

GAIA-X 4 Future Mobility



Schlussbericht GAIA-X 4 AGEDA

Robert Bosch GmbH
19S22004I



Finanziert von der
Europäischen Union
NextGenerationEU

Gefördert durch:



Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

Dokumenteigenschaften

Titel	Schlussbericht
Betreff	GAIA-X 4 AGEDA
Vorhabenbezeichnung	Verbundprojekt: GAIA-X 4 AGEDA - Anforderungen und Anwendung von GAIA-X im Edge-Device Automobil
Laufzeit des Vorhabens	01.10.2022 – 31.12.2025
Berichtszeitraum	01.10.2022 – 31.12.2025
Firma / Institut	Robert Bosch GmbH
Erstellt von	Michael Schulz
Beteiligte	Arthur Ehlert, Yannick Fischer, Erik Walossek
Geprüft von	Armin Karle
Freigabe von	Armin Karle
Datum	19.02.2026
Version	V2
Dateipfad	

Inhalt

Abbildungsverzeichnis	4
Tabellenverzeichnis	5
Abkürzungsverzeichnis	5
1 Zusammenfassung.....	7
2 Kurzdarstellung zu Projektrahmen und Ablauf	7
2.1 Zusammenfassung der AGEDA-Projektidee, sowie der Gesamtzielsetzung und Aufgabenstellung.....	7
2.1.1 Projektidee.....	7
2.1.2 Aufgabenstellung und Arbeitspakete	9
2.2 Voraussetzungen des Vorhabens.....	10
2.3 Planung und Ablauf des Vorhabens	12
2.4 Beitrag der Robert Bosch GmbH - Wissenschaftlich-technischer Stand und Aufgabenstellung	12
2.5 Zusammenarbeit mit Dritten	13
3 Durchgeführte Arbeiten und erreichte Ergebnisse	13
3.1 Teilprojekt 1: Anwendungen.....	13
3.1.1 AP 1.1: Ausgestaltung der Anwendungsfälle.....	13
3.1.2 AP 1.3: Demonstrator-Implementierung.....	14
3.1.3 AP 1.4: Verifikation und Validierung	25
3.2 Teilprojekt 2: Bruchlose Integration von GAIA-X Gestaltungsprinzipien in innovative Fahrzeugarchitekturen	25
3.2.1 AP 2.1: Software- und Hardware-Architektur und AGEDA Komponenten.....	25
3.2.2 AP 2.3: Dynamische Rekonfiguration	30
3.3 Teilprojekt 3: Methoden, Werkzeuge und Zertifizierung	31
3.3.1 AP 3.2: Betrieb	31
4 Wichtigsten Positionen des zahlenmäßigen Nachweises	52
5 Notwendigkeit und Angemessenheit der geleisteten Arbeit	52
6 Voraussichtlicher Nutzen und Verwertbarkeit der Ergebnisse	53
7 Erfolgte oder geplante Ergebnisverbreitung.....	53
8 Literaturverzeichnis.....	53

Abbildungsverzeichnis

Abbildung 1: Projektstruktur AGEDA - Arbeitspakete und fachliche Beiträge.....	9
Abbildung 2: AGEDA Arbeitspakete im Überblick	10
Abbildung 3: Projektpartner und assoziierte Partner des AGEDA Förderprojekts	11
Abbildung 4: Messestand vom AGEDA-Konsortium bei der Hannover Messe in Zusammenarbeit mit anderen GAIA-X Projekten	15
Abbildung 5: Übersicht der Demonstration und dem Zusammenspiel zwischen moveID, AGEDA und NeMO	16
Abbildung 6: Ausschnitt aus der CARLA Simulation und dem Credential Interaction Flow.....	18
Abbildung 7: Eingesetztes Versuchsfahrzeug (VW Golf VII Variant) für die CVC-L4 Demonstration	18
Abbildung 8: Vereinfachte Systemarchitektur des Fahrzeuges mit integriertem AGEDA Framework	19
Abbildung 9: Modifiziertes Kombiinstrument mit der Möglichkeit, Inhalte frei zu bespielen.....	19
Abbildung 10: Implementierte Fehlermeldung zur realitätsnahen Umsetzung des CVC-L4 Szenarios	19
Abbildung 11: Bosch Softwarekomponenten - Demo 2025.....	19
Abbildung 12: SOVD-Demo 2024 Setup	21
Abbildung 13: Browser-Benutzeroberfläche des AVL DiTEST Remote-Clients-Web-Client	21
Abbildung 14: Workflow und Bestandteile des Table Demonstrators.....	22
Abbildung 15: Remote SOVD-Client vom Table Demonstrator.....	23
Abbildung 16: Foto vom Table Demonstrator vom Abschlussevent in Lippstadt Nov. 2025.....	23
Abbildung 17: Ablauf der Demo	24
Abbildung 18: Ablauf der Demo	24
Abbildung 19: Präsentation der finalen Projektergebnisse von AVL DiTest und Robert Bosch GmbH zum Thema Monitoring und Update am Versuchsfahrzeug der Robert Bosch GmbH.....	25
Abbildung 20: Übersicht über die Geschäftsbeziehungen der involvierten Partner	27
Abbildung 21: Sequenz Diagramm Zugangsberechtigung Use Case	28
Abbildung 22: Sequenzdiagramm Zugangsberechtigung Use Case	30
Abbildung 23: Auszug der Übersicht zum Thema Monitoring und Logging von der Confluence-Seite (Link)	34
Abbildung 24: ASAM SOVD Schema (Quelle: https://www.asam.net/standards/detail/sovd/)	35
Abbildung 25: SOVD-Server als AGEDA Workload	36
Abbildung 26: SOVD-Server als AGEDA Building Block	38
Abbildung 27: Verbindung zum SOVD-Server mittels dedizierten Interface	39
Abbildung 28: ADR zur SOVD-Integration	41
Abbildung 29: Finale AF Architektur Monitoring	41
Abbildung 30: SOVD-Architektur für AGEDA.....	42
Abbildung 31: Browser-Benutzeroberfläche des AVL DiTEST SOVD Remote Clients.....	44
Abbildung 32: Übersicht CVC-Breakdown-Workload.....	44
Abbildung 33: Teilnehmerübersicht Traffic-Awareness-Service	45
Abbildung 34: Momentaufnahme Birds-Eye-GUI.....	45
Abbildung 35: Code-Snippet SOVD-Connector	47
Abbildung 36: Update-Schritte bei Live-Demonstration	49
Abbildung 37: Update Konzept Umriss	50
Abbildung 38: Ablauf eines Updates	51

Tabellenverzeichnis

Tabelle 1 Beschreibung der Meilensteine	12
Tabelle 2: Übersicht über die GAIA-X Ziele für den M36 mit Bosch Beteiligung (Auszug aus Confluence)	26
Tabelle 3: Vor- und Nachteile SOVD als AGEDA Workload	37
Tabelle 4: Vor- Nachteile SOVD als AGEDA Building Block.....	38
Tabelle 5: Vor- und Nachteile SOVD außerhalb des AF.....	39
Tabelle 6: Notwendige Ressourcen für eine SOVD-Anbindung	46
Tabelle 7: Lister der Update Use-Cases.....	48

Abkürzungsverzeichnis

AAOS	Android Automotive OS
ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance Systems
AGEDA	Anforderungen und Anwendung von GAIA-X im Edge-Device Automobil
AF	AGEDA Framework
AP	Arbeitspaket
APA	Automated Parking Assist
ASIL	Automotive Safety Integrity Level
B2B	Business-to-Business
B2C	Business-to-Consumer
BLE	Bluetooth Low Energy (angenommen, da in Kontext von Lokalisierung steht)
BMWE	Bundesministerium für Wirtschaft und Energie
CAN	Controller Area Network
CCU	Communication Control Unit
CIP	Cockpit Integration Plattform
CPU	Central Processing Unit
CVC	Collective Vision and Control
DID	Decentralized Identifier
E2E	End-to-End
ECU	Electronic Control Unit
E/E	Elektrik/Elektronik (angenommen, da "E/E-Architekturen" verwendet wird)
FPK	Frei-Programmierbares Kombiinstrument
FW	Framework
GPU	Graphics Processing Unit
GPS	Global Positioning System
HMI	Human-Machine Interface
IoT	Internet of Things
JSON	JavaScript Object Notation
KVM	Keyboard, Video, Mouse
LTS	Long Term Support
MCU	Microcontroller Unit
MRR	Mid-Range Radar
MQTT	Message Queuing Telemetry Transport
NPU	Neural Processing Unit
OCM	Organization Credential Manager
ODS	Object Detection Service
OEM	Original Equipment Manufacturer
ÖgP	Öffentlich gefördertes Projekt
PK	Perfectly Keyless
QNX	Name eines Echtzeit-Betriebssystems

RAM	Random Access Memory
SAE	Society of Automotive Engineers
SdV	Software-defined Vehicle
SDP	Service Data Point
SoC	System-on-a-Chip
SOVD	Software Over-the-air
SSI	Self-Sovereign Identity
SSH	Secure Shell
TAS	Traffic Awareness Service
TP	Teilprojekt
TRL	Technology Readiness Level
UID	Unique Identifier
VC	Verifiable Credential
VCPU	Virtual Central Processing Unit
VM	Virtual Machine
VP	Verifiable Presentation
VSS	Vehicle Signal Specification
VV	Verifikation und Validierung
V2X	Vehicle-to-Everything
WP	Work Package

1 Zusammenfassung

Der vorliegende Schlussbericht zum BMW-geförderten Projekt AGEDA („Anforderungen und Anwendung von GAIA-X im Edge-Device Automobil“) fasst den Beitrag der Robert Bosch GmbH zusammen.

Der Bericht ist wie folgt gegliedert: Das nachfolgende Kapitel enthält eine Kurzdarstellung zu Projektrahmen und Ablauf gem. NKBF 98, Anlage 2, Teil I. Darin wird mit einer kurzen Einführung in die AGEDA-Projektidee begonnen, sowie die Zielsetzung und Aufgabenstellung des gesamten Projektes eingeleitet. Dann folgen Informationen zu den Voraussetzungen und dem Ablauf des Projekts. Im Anschluss werden der Stand der Technik, sowie die partnerspezifischen Aufgabenschwerpunkte der Robert Bosch GmbH nach Arbeitspaketen gegliedert dargestellt.

Die daran anschließenden Kapitel enthalten eine ausführliche Darstellung gem. NKBF 98, Anlage 2, Teil II. Dort werden die von der Robert Bosch GmbH durchgeführten Arbeiten und erzielten Ergebnisse, gegliedert nach den Arbeitspaketen und Komponenten, im Detail beschrieben. Im Schlussteil des Berichtes werden die wichtigsten Positionen zum zahlenmäßigen Nachweis, Informationen zum Verwertungsplan und den Erfolgsaussichten, sowie zu durchgeführten Maßnahmen zur Ergebnisverbreitung zusammengefasst.

2 Kurzdarstellung zu Projektrahmen und Ablauf

2.1 Zusammenfassung der AGEDA-Projektidee, sowie der Gesamtzielsetzung und Aufgabenstellung

2.1.1 Projektidee

Heutige Fahrzeugarchitekturen basieren in der Regel noch auf dezentralen oder domänenzentrierten Konzepten, die stark auf einzelne Steuergeräte fokussieren. Neue Anwendungen und Dienste, die gegebenenfalls nur temporär genutzt oder benötigt werden, lassen sich mit diesen Architekturen nur sehr schwer realisieren. Dies gilt insbesondere, wenn externe Datenquellen und Services involviert sind, um beispielsweise einen bidirektionalen Informationsaustausch zwischen Fahrzeugen und deren Infrastruktur zu ermöglichen. Die dynamische Anpassung an neue Gegebenheiten, wie die Verfügbarkeit neuer Datenquellen oder die Nutzung spezifischer Fahrzeugfunktionen durch Drittanbieteranwendungen, ist hier kaum oder gar nicht realisierbar.

Vor diesem Hintergrund setzte sich das Projekt AGEDA zum Ziel, eine Software-Architektur im Edge-Device Fahrzeug zu entwickeln, die die Konzepte der GAIA-X-Initiative nutzte und somit "by-design" datengetriebene Anwendungen sowie die dynamische Anpassung eines Fahrzeugs mit seinen Funktionen über den gesamten Lebenszyklus ermöglichte.

Das Projekt betrachtete und realisierte Anwendungsfälle, in denen durch die Vernetzung des Fahrzeugs – insbesondere auch durch die Nutzung sicherheitskritischer Daten aus dem Fahrzeug und fahrzeugfremder Daten für sicherheitskritische Funktionen – Mehrwerte für Fahrzeughersteller, Fahrzeugnutzer und das Mobilitätssystem insgesamt entstanden.

Die im Projekt entwickelte Fahrzeugarchitektur unterstützte nativ die Vernetzung aller Fahrzeugfunktionen mit Cloud-basierten Diensten. Sie ermöglichte es, auch nach Auslieferung eines Fahrzeugs zusätzliche, völlig neuartige Anwendungen zu realisieren, die eine dynamische Anpassung des Fahrzeugs an neue Gegebenheiten erlaubten und somit einen entscheidenden Beitrag zur schnellen Umsetzung der Mobilitätswende leisteten. Neue Entwicklungsmethoden und Zulassungsverfahren ermöglichten eine rasche Transformation der Mobilität durch die Umsetzbarkeit auch hochautomatisierter Fahrfunktionen.

Schnittstellen vom Fahrzeug und in das Fahrzeug hinein – nicht nur für den Datenaustausch, sondern auch für den Austausch von Steuerungsbefehlen – ermöglichten völlig neuartige Geschäftsmodelle und damit

Mobilitätsdienstleistungen. So eröffnete die neuartige Fahrzeugarchitektur völlig neue Märkte, trug zur Schaffung von Arbeitsplätzen bei und sicherte letztlich die Wirtschaftsstandorte Deutschland und Europa. Die Nutzung von GAIA-X als Basis für den Datenaustausch stellte dabei die europäische Datensouveränität sicher.

Die Kerninnovation des Projektes war eine Referenzimplementierung einer neuartigen Fahrzeugarchitektur inklusive beispielhafter Umsetzungen vernetzter Mobilitätsanwendungen, die im Rahmen einer Abschlussveranstaltung öffentlich demonstriert wurden.

Konkret verfolgte das Projekt AGEDA die nachfolgend vorgestellten wissenschaftlichen / technischen Arbeitsziele:

- **Entwicklung einer Fahrzeug-IT-Architektur, die nativ eine Vernetzung mit fahrzeugexternen Umgebungen ermöglicht**

Bei diesem Ziel geht es darum, dass die in Fahrzeugen genutzte Software so implementiert wird, dass ein dynamischer (d. h. softwarebasierter) Zugriff auf alle Fahrzeugfunktionen möglich werden soll. Die Motivation hinter diesem Ziel ist die Tatsache, dass Sensoren, Aktoren und Fahrzeugfunktionen Potenziale bieten, deren Umfang zur Entwicklungszeit des Fahrzeugs nicht vollständig abgeschätzt, im Betrieb des Fahrzeugs aber dynamisch über Software-Updates gehoben können werden soll.

- **Entwicklung von Konzepten für die dynamische Rekonfiguration von Software-Funktionen im Fahrzeug**

Bei diesem Ziel geht es darum, die Potenziale der in einem Fahrzeug verbauten Recheneinheiten dynamisch nutzen zu können. Dies ist insbesondere erforderlich, um die Resilienz softwarebasierter Funktionen im Fahrzeug zu erhöhen, aber auch um nach Funktionserweiterungen die Sicherheit des Fahrzeugs und den effizienten Betrieb sicherstellen zu können.

- **Demonstration der Potenziale einer neuartigen Fahrzeug-IT-Architektur**

Die Referenzimplementierung der neuartigen Fahrzeugarchitektur soll im Rahmen des Projektes anhand konkreter Anwendungsfälle das Potenzial sowohl der fahrzeuginternen aber auch der fahrzeugexternen Vernetzung darstellen. Insbesondere dynamische softwarebasierte Funktionserweiterungen werden zukünftig einen großen Beitrag zur Wertschöpfung, zur Mobilitätswende aber insbesondere auch zur Wettbewerbsfähigkeit von OEMs und Zulieferern leisten. Die dargestellten Anwendungsfälle dienen dazu, diese Potenziale exemplarisch darzustellen.

- **Entwicklungs-, Betriebs- und Zertifizierungsprozesse**

Mit dem Umbau der Fahrzeug-IT-Architektur, insbesondere zur rein softwarebasierten Umsetzung auch sicherheitskritischer Funktionen, kommt auch der Bedarf nach entsprechenden Entwicklungs-, Betriebs- und Zertifizierungsprozessen für Entwicklung von Anwendungen auf Basis dieser neuen Architektur zum Tragen. Im Rahmen des Projektes werden bestehende Ansätze aus anderen Vorhaben (VV-Methoden, StepUp!CPS, PEGASUS, SetLevel, etc.) dahingehend betrachtet und erweitert, dass sie auch die Umsetzung software- und cloud-basierter sicherheitskritischer Funktionen im Fahrzeug unterstützen und diese dann eine entsprechende Zulassungsfähigkeit erhalten.

Diese zuvor genannten Punkte spiegeln sich in der nachfolgend dargestellten Projektstruktur (siehe Abbildung 1) wider.

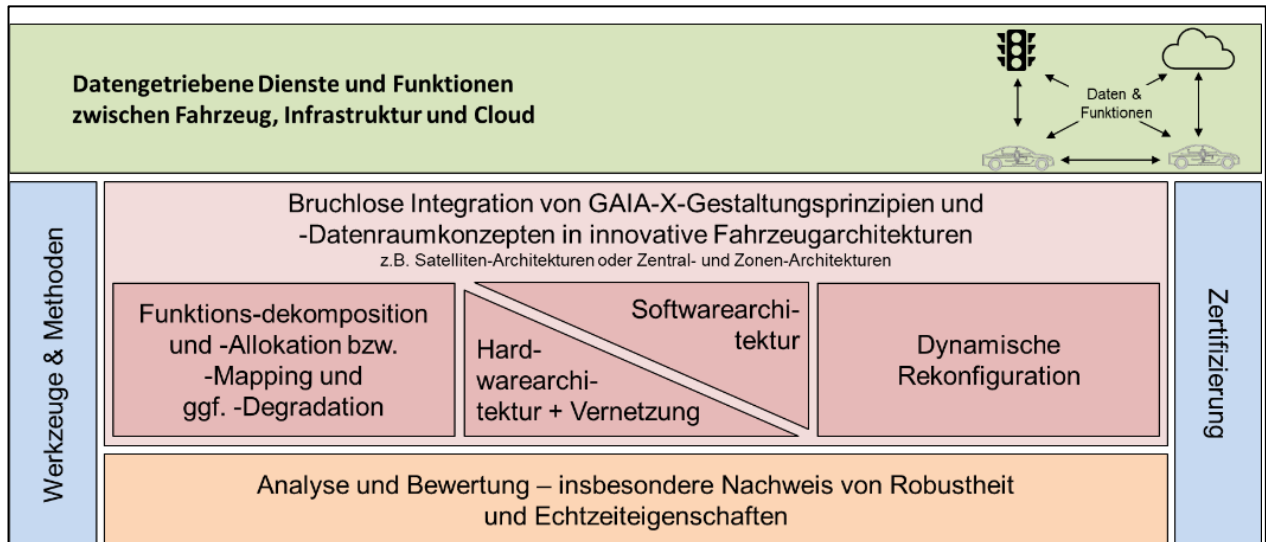


Abbildung 1: Projektstruktur AGEDA - Arbeitspakete und fachliche Beiträge

Diese Projektstruktur diente dazu, aus den Anwendungsfällen möglicher datengetriebener Dienste und Funktionen zwischen Fahrzeug, Infrastruktur und Cloud heraus die notwendigen technischen Konzepte an eine solche Fahrzeugarchitektur abzuleiten. Basierend auf dieser Ableitung wurden dann die einzelnen technischen Komponenten der Fahrzeugarchitektur, einschließlich ihres dynamischen Verhaltens, in Form einer Referenzimplementierung umgesetzt und anschließend anhand konkreter Anwendungsfälle, wie sie in TP 1 dargestellt wurden, instanziiert und demonstriert.

Das Projekt setzte dabei insbesondere auf einen inkrementellen Ansatz, der auf eine frühzeitige Realisierung greifbarer Teilergebnisse abzielte, um möglichst frühzeitig Feedback auch aus den Implementierungs- und Erprobungsphasen des Projektes in die weitere Entwicklung einzubeziehen.

2.1.2 Aufgabenstellung und Arbeitspakete

Das Projekt nahm datengetriebene Dienste und Funktionen, die zukünftig im After-Sales-Market oder auch als Ausrüstung von Fahrzeugen angeboten werden sollten, als Leitplanke, entlang derer die Anforderungen und Konzepte für eine neue Fahrzeugarchitektur entwickelt wurden. Die zentrale Aktivität im Projekt war dann die Entwicklung der einzelnen Bestandteile einer solchen Architektur. Gleichzeitig wurden, um insbesondere die industriellen aber auch die Zulassungsprozesse zu unterstützen und damit eine einfache Marktzugänglichkeit zu ermöglichen, auch die notwendigen Entwicklungs-, Zulassungs- und Betriebskonzepte, -werkzeuge und -methoden entwickelt. Über den Aufbau von Demonstratoren wurden dann sowohl die entwickelten Architekturkonzepte und -bestandteile als auch die zuvor genannten Methoden und Werkzeuge evaluiert. Nachfolgend ist in Abbildung 2 die Arbeitspaketstruktur mit den vier Teilprojekten „Anwendungen“, „Bruchlose Integration von GAIA-X-Gestaltungsprinzipien in innovative Fahrzeugarchitekturen“, „Methoden, Prozesse, Werkzeuge“ und „Projektkoordination, Vernetzung und Ergebnisverbreitung“ dargestellt. Insgesamt unterteilte sich das AGEDA Projekt in 13 Arbeitspakete.

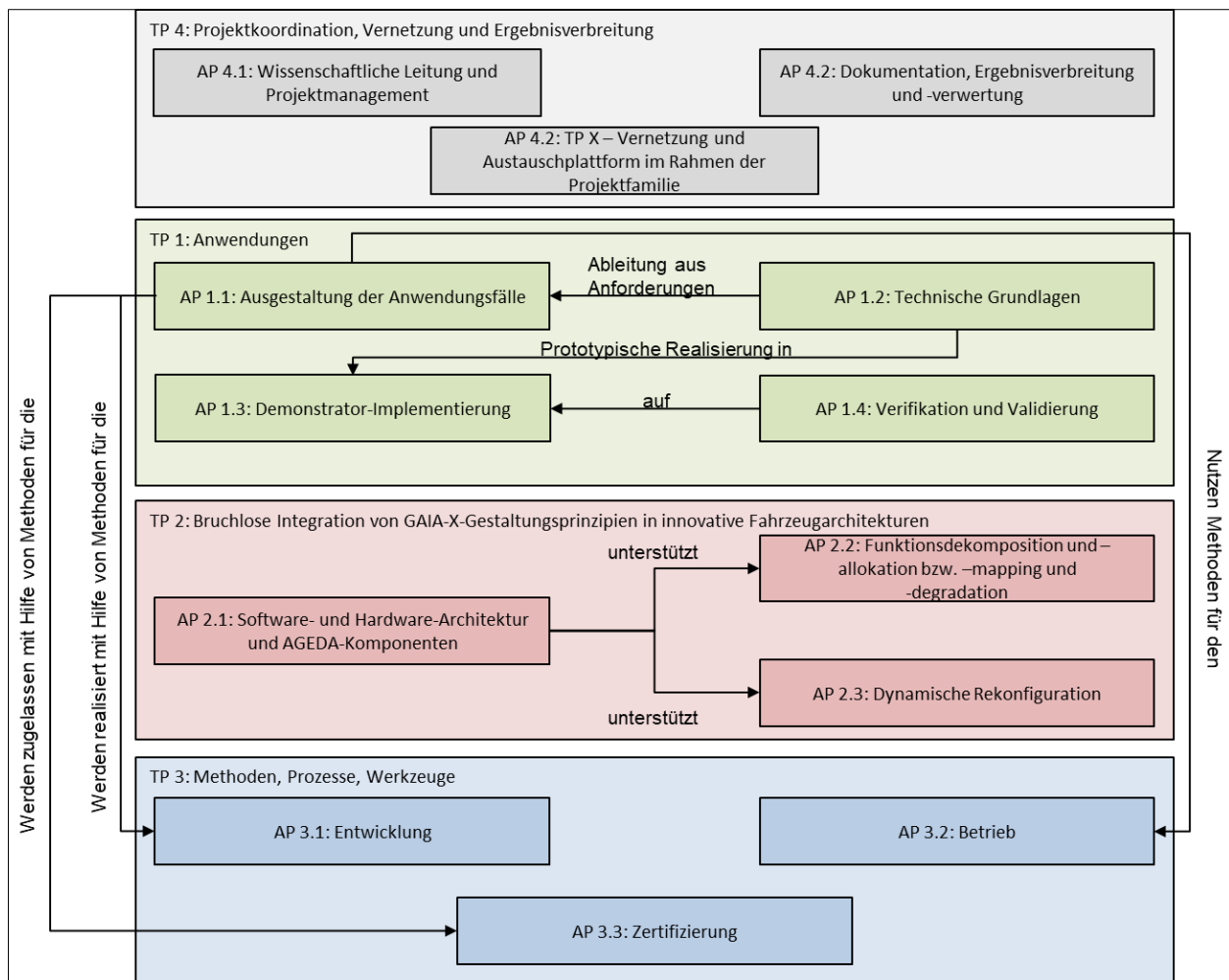


Abbildung 2: AGEDA Arbeitspakete im Überblick

2.2 Voraussetzungen des Vorhabens

Das Projekt AGEDA wurde in einem Verbund aus Automobilherstellern, Zulieferern, öffentlichen Einrichtungen, Forschungseinrichtungen und KMUs bearbeitet. Das Gesamtvolumen betrug 27,3 Mio €, mit einer Fördersumme des BMWV von bis zu 14,2 Mio €. Die Projektkoordination von AGEDA oblag der Hella GmbH & Co. KGaA. Die Projektpartner waren:

- Hella GmbH & Co. KGaA
- Volkswagen AG
- Robert Bosch GmbH
- Continental Automotive Technologies GmbH
- AVL DiTEST GmbH
- AVL Software & Functions GmbH
- Elektrobit Automotive GmbH
- IAV GmbH Ingenieurgesellschaft Auto und Verkehr
- ITK Engineering GmbH
- SUSE Software Solutions Germany GmbH
- Vodafone Group Service GmbH
- Urban Software Institute GmbH
- embeteco GmbH & Co. KG
- Institut für Angewandte Informatik (InfAI) e.V.

- Hochschule Hamm-Lippstadt
- Deutsches Zentrum für Luft- und Raumfahrt e. V.
- Otto GmbH & Co. KG (associated partner)
- Forschungsvereinigung Automobiltechnik (FAT) e.V. (associated partner)
- ARM Germany (associated partner)
- T-Systems International GmbH (associated partner)
- Mercedes-Benz AG (associated partner)

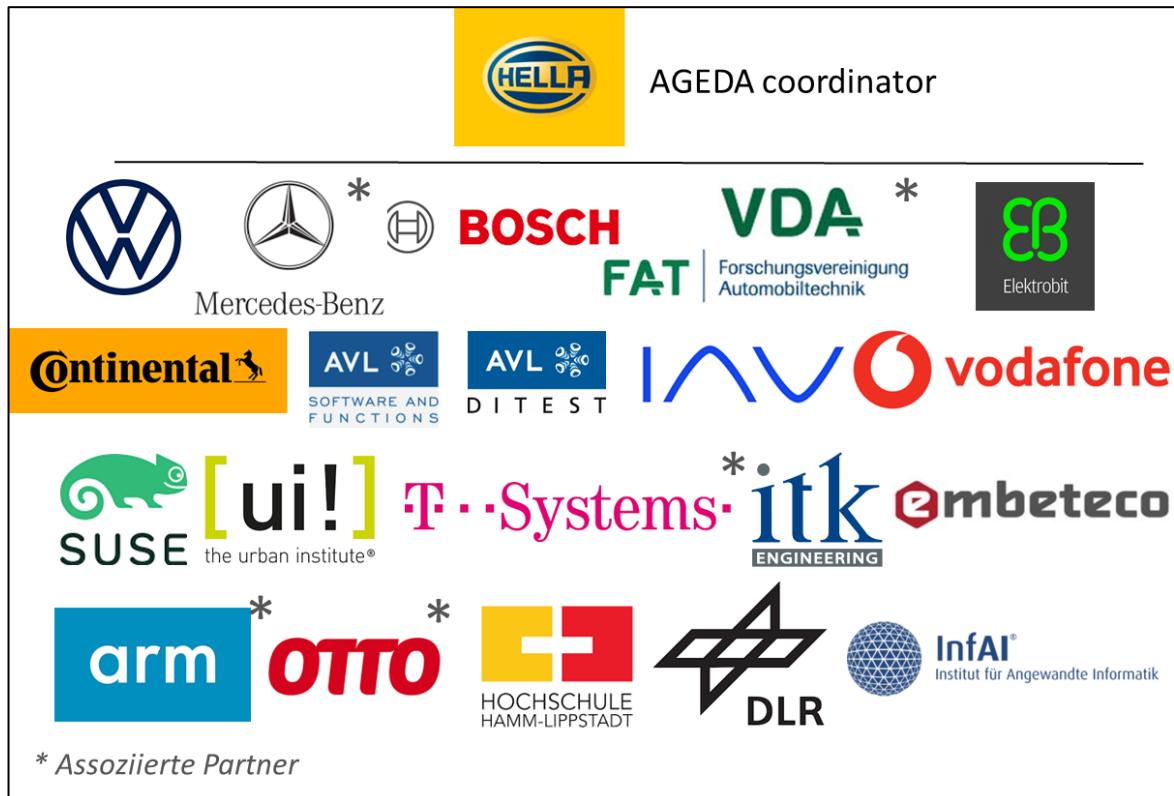


Abbildung 3: Projektpartner und assoziierte Partner des AGEDA Förderprojekts

2.3 Planung und Ablauf des Vorhabens

Das Projekt startete am 01.10.2022 und endete nach einer viermonatigen Verlängerung nach einer Laufzeit von 39 Monaten am 31.12.2025.

Das Projekt AGEDA war, wie in Kapitel 2.1.2 erläutert, in vier Teilprojekte und insgesamt 13 Arbeitspakete gegliedert. Zeitlich wurden weiterhin folgende Projekt Meilensteine definiert:

Tabelle 1 Beschreibung der Meilensteine

Meilenstein	Monat ab Projektbeginn	Ziele
Meilenstein 1	12	<ul style="list-style-type: none"> • Fahrzeugarchitektur auf TRL 3 • 1 Datenquelle angebunden • 1 GAIA-X Konzept integriert • 1 Design-Kriterium festgelegt • 1 Geschäftsmodell identifiziert
Meilenstein 2	24	<ul style="list-style-type: none"> • Fahrzeugarchitektur auf TRL 4 • 1 weitere Datenquelle angebunden • 1 weiteres GAIA-X Konzept integriert • 1 weiteres Geschäftsmodell identifiziert • Nachrüsten einer Funktion ist möglich
Meilenstein 3	36	<ul style="list-style-type: none"> • Fahrzeugarchitektur auf TRL 5 • Alle Ziele aus Abschnitt 2.4

Die Zeitplanung wurde dabei mit kleineren Abweichungen eingehalten und der Projektfortschritt in den zugehörigen Berichten dokumentiert. Die Zwischenpräsentation des Projektes fand am 14.11.2024 in Wolfsburg bei der Volkswagen AG sowie die Abschlusspräsentation am 22.10.2025 an der Hochschule Hamm-Lippstadt in Lippstadt statt.

2.4 Beitrag der Robert Bosch GmbH - Wissenschaftlich-technischer Stand und Aufgabenstellung

Der wissenschaftlich-technische Stand, an den von der Robert Bosch GmbH zu Projektbeginn angeknüpft wurde, speist sich im Wesentlichen aus den umfangreichen Vorarbeiten im Bereich Datenerhebung aus den Fahrzeugen, Datenverarbeitung im Backend, over-the-air SW Aktualisierung sowie Erkenntnisse aus den komplementären Förderprojekten SofDCar, Move-ID und Catena-X in AGEDA ein.

Neben den internen Vorarbeiten bei Bosch speiste sich ein umfangreicher Teil des Stands der Technik aus den Erkenntnissen öffentlich geförderter Projekte, die vor AGEDA begannen. Dort wurde, teilweise unter Bosch-Beteiligung, ein Wissensstand aufgebaut, der einerseits neue Fahrzeugarchitekturen im Kontext von Software-definierten Fahrzeugen (SdV) behandelte und andererseits weiteren Forschungs- und Innovationsbedarf für die in AGEDA zu betrachtende Gebiete aufzeigte.

Die Robert Bosch GmbH entwickelte im Rahmen des SofDCar ÖgP eine Software-Architektur für Domänen-Steuergeräte (ECUs) beziehungsweise Zentralcomputer eines Fahrzeugs. Bestandteile dieser Software-Architektur oder deren Implementierung, insbesondere mit Fokus auf datengetriebene Entwicklung und Betrieb, wurden in AGEDA in anderweitig entwickelten Architekturen erprobt, weiterentwickelt, integriert und betrieben. Des Weiteren brachte Bosch seine Kompetenz in der Flexibilisierung und der State-of-the-Art-Updatefähigkeit der tief eingebetteten ECUs ein, um die Updateprozeduren in Fahrzeugen bei gleichbleibender oder sogar erhöhter Erfüllung von Safety- und Security-Anforderungen zu verkürzen.

Die genaue Abgrenzung der bereits laufenden bzw. abgeschlossenen Projekte zu AGEDA wurde in der Vorhabenbeschreibung der Verbundpartner genauer dargelegt.

Die Robert Bosch GmbH übernahm die Leitung des Arbeitspaketes 3.2 (TP3.2) und brachte im AGEDA-Projekt vorrangig ihr Know-how im Bereich Monitoring und Update ein. Bosch definierte zusammen mit den Partnern entsprechende Anforderungen und vertrat diese bei der Entwicklung des AGEDA-Frameworks. Darüber hinaus entwickelte Bosch in enger Zusammenarbeit mit den Projektpartnern passende Update- und Monitoring-Konzepte, evaluierte diese und implementierte sie für den Einsatz im Realfahrzeug sowie im Table Demonstrator. Im Rahmen der "Collective Vision and Control Demo" war Bosch nicht nur für die Monitoring- und Update-Szenarien verantwortlich, sondern beteiligte sich auch mit einem Versuchsfahrzeug an der Demonstration.

Im GAIA-X-Kontext beschäftigte sich Bosch intensiv mit digitalen Identitäten, Verifiable Credentials (VCs) / Presentations sowie der End-to-End (E2E) GAIA-X-Architektur. In diesem Zusammenhang wurden verschiedene Anwenderszenarien (User Journeys) für die Nutzung digitaler Identitäten und digitaler Nachweise (Verifiable Credentials) entwickelt.

2.5 Zusammenarbeit mit Dritten

Neben dem fachlichen Austausch mit den anderen Verbundpartnern wurde auch der Fortschritt von Wissenschaft und Technik außerhalb von AGEDA während der Projektlaufzeit kontinuierlich beobachtet und analysiert. Dies geschah einerseits im Rahmen von Bosch-internen Veranstaltungen zum Austausch über wissenschaftlichen Themen und andererseits durch die Verfolgung der wissenschaftlichen Literatur. Weiterhin erfolgte die Teilnahme an den Future Mobility Projektfamilie Treffen, der Hannover Messe sowie der IAA. Im Rahmen der Hannover Messe erfolgte ein intensiverer Austausch mit dem Schwesterprojekt MoveID sowie dem Projekt NeMo.bil.

3 Durchgeführte Arbeiten und erreichte Ergebnisse

3.1 Teilprojekt 1: Anwendungen

3.1.1 AP 1.1: Ausgestaltung der Anwendungsfälle

Bosch hat in enger Zusammenarbeit mit den Partnern im Arbeitspaket 1.1 die Use Cases ausgearbeitet und evaluiert. Dabei konzentrierte sich Bosch auf die Update & Monitoring Use Cases. Darüber hinaus brachte das Unternehmen seine Expertise auch in die anderen Use Cases ein, unter anderem durch die Bereitstellung eines Versuchsfahrzeugs und die Implementierung von Funktionen für die CVC-L4 Demo im Projektverlauf.

Nachfolgend sind die initialen Use-Cases für die CVC Demo aufgeführt:

- **CVC-L2-SC1-UC13** - Detect & reject malicious Services on Vehicles Edge
Dieser Use-Case umfasst die Überwachung der Aktivität von Benutzern und Services. Wenn eine böswillige Aktivität erkannt wird, reagiert das Fahrzeug entsprechend. Hierbei wurden zwei Schlüsselszenarien beispielhaft definiert:
 - Ein Service versucht Zugriff auf Fahrzeugsignale zu bekommen und bekommt diesen Zugriff, obwohl keine Berechtigung vorhanden ist
 - Ein Service verursachte falsche Nachrichten und belastet die Cloud mit Spam, was zu einer Beeinträchtigung der Leistung
- **CVC-L2-SC1-UC14** - In-Vehicle Service is executing incorrectly & stopped
Ein Service verhält sich inkorrekt gegenüber seiner Funktionsbeschreibung (z.B. erhöhter Speicherbedarf). Das inkorrekte Verhalten wird erkannt und der Service gestoppt.
- **CVC-L2-SC1-UC15** - In-vehicle reconfiguration due to cloud connection quality degradation
Für die notwendigen Services stehen nicht alle Ressourcen (z.B. Speicher, Netzwerkverbindung) zur Verfügung, was eine Priorisierung und dynamische Ressourcenzuordnung erforderlich macht.

- **CVC-L3-SC1-UC14:** Monitoring of Status of AGEDA-FW-Node and Workloads via SOVD-Server-Client

Analog wurden Use Cases für das Update Szenario entwickelt:

- **CVC-L4-SC1-UC16** - Update of workloads
Ausführung eines Updates eines verfügbaren Services automatisch, nach User Bestätigung oder mit Hilfe des SOVD Servers.
- **Update of a Workload – Automatic**
Eine automatische Aktualisierung eines Workloads wird während des Systemstarts (oder danach) ausgelöst, ohne manuelle Benutzerinteraktion.
- **Update of Workload - with user interaction** (user needs to approve)
Ein laufender Workload A wird manuell aktualisiert, entweder über eine Versionsübersicht oder durch Bestätigung eines Pop-ups. Besondere Beachtung findet die erneute Zustimmung zu Vertragsbedingungen bei Änderungen.
- **Update of a Workload - On Demand via SOVD by technician**
Ein Techniker löst manuell die Aktualisierung eines spezifischen Workloads (Workload A) über eine im Fahrzeug verfügbare SOVD-Workload-Benutzeroberfläche aus.
- **Update of workload while AGEDA FW and workload are running**
Laufender Workload wird automatisch auf eine neuere Version aktualisiert, während das AGEDA-Framework aktiv ist und ohne direkte Benutzerinteraktion oder Systemneustart.
- **Update of workload - critical bug in workload**
Ein kritischer Fehler im Workload wird automatisch erkannt und veröffentlicht (z.B. im Federated Catalogue). Das AGEDA Framework erhält die Information und stoppt den Service automatisch.
- **Update AGEDA FW core component during startup**
Kernelemente des AGEAD Frameworks werden nach / während des Startprozess aktualisiert
- **Update of plugin during startup**
Plugins des AGEDA Frameworks werden nach / während des Startprozess aktualisiert

Die definierten Use-Cases dienen als Grundlage für die Konzepte und der Implementierung der Monitoring und Update Use-Cases, als auch für die Ausbaustufen des Collective Vision & Control Szenarios.

3.1.2 AP 1.3: Demonstrator-Implementierung

Dieses Kapitel widmet sich den Beiträgen von Bosch zur Demonstrator-Implementierung. Bosch war in enger Abstimmung mit AP1.1 maßgeblich an der Gestaltung der Drehbücher für die Demonstrationen beteiligt und unterstützte das Konsortium bei der Implementierung und Inbetriebnahme der dafür benötigten Systemkomponenten.

Besonders hervorzuheben sind der für die Hannover Messe entwickelte Demonstrator, der Table Demonstrator sowie das Versuchsfahrzeug zur Präsentation der Projektergebnisse bei Halbzeit- und Abschlussevents. Darüber hinaus war das von Bosch bereitgestellte und aufgebaute Versuchsfahrzeug ein entscheidendes Element für die CVC L4 Demonstration am Toastmannplatz in Braunschweig.

3.1.2.1 Demonstrator Hannover Messe

Ein großer Meilenstein während der Projektlaufzeit war die Präsentation der erreichten Ergebnisse auf der Hannover Messe. Diese Industriemesse mit internationaler Bedeutung fand vom 22.04. bis zum 26.04.2024 statt. Im Mittelpunkt in diesem Jahr standen künstliche Intelligenz, klimaschonende Produktion, Lösungen für die Energiewende und das Thema Wasserstoff als Energieträger.

Messestand

Die Gaia-X 4 Future Mobility Projektfamilie hatte 72 qm Fläche auf dem Gaia-X Community Stand in der Halle 8 (F25). Für jedes der 6 Schwesterprojekte stand ein 40“ Monitor mit Stehle zur Verfügung. Für AGEDA wurde hier der Collective Vision and Control (CVC) Demonstrator, welcher in Zusammenarbeit mit den Projekten NeMo.bil und MoveID entstand, gezeigt. Um auch die Projektfolien präsentieren zu können, wurde der vorhandene Monitor und die Tastatur über einen KVM-Switch mit zwei Rechnern verbunden, der es ermöglichte, beliebig zwischen den Folien und der Präsentation zu wechseln (siehe Abbildung 4).

Eine permanente Beteiligung am Standdienst durch sämtliche am Demonstrator mitwirkende Partner war schon aufgrund der begrenzten Platzverhältnisse am Stand nicht sinnvoll, so dass im Vorfeld die Teilnahme der verschiedenen Projektpartner koordiniert wurde. Bosch beteiligte sich am Dienstag den 23.04. und am Donnerstag den 25.04. ganztägig von 9-18 Uhr am Standdienst. Nach einer kurzen Einweisung am realen AGEDA Demonstrator wurde der CVC Use Case diversen interessierten Messebesuchern vorgeführt. Als Eyecatcher diente das durch das Projekt NeMo.bil (Neue Mobilität Paderborn) ausgestellte prototypische Fahrzeug, das in Zukunft bis zu 4 Personen vernetzt, klimaschonend und benutzerorientiert befördern können soll.

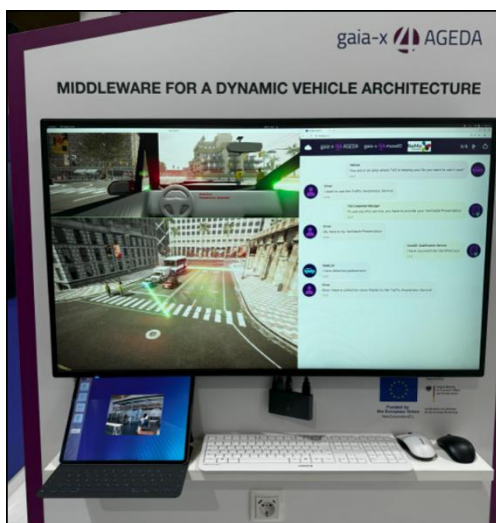


Abbildung 4: Messestand vom AGEDA-Konsortium bei der Hannover Messe in Zusammenarbeit mit anderen GAIA-X Projekten

In Vorbereitung zur Messe koordinierte Bosch die Zusammenarbeit innerhalb einer Arbeitsgruppe der Projektpartner aus AGEDA und MoveID und leitete die wöchentlichen Regelmeetings im Zeitraum vom 05.02. bis 29.04.24. Hier wurden die konzeptuellen und technischen Anforderungen für die Demo erfasst und eine User Story definiert. Daraus wurde im Anschluss die Systemarchitektur definiert und die identifizierten Rollen einzelnen verantwortlichen Personen zugeordnet. Die wöchentlichen Meetings ermöglichten eine intensive Zusammenarbeit und ein offenes Tracking der einzelnen den Projektpartnern zugeordneten Aufgaben.

Aufbau des Demonstrators

Der Messedemonstrator soll zeigen, dass mit Hilfe der AGEDA Middleware ein Fahrzeug ein Teil von GAIA-X sein kann. Vertrauenswürdige Objektdaten werden durch einen Object Detection Service (ODS) einem Traffic Awareness Service (TAS) zur Verfügung gestellt, der diese dann anderen Verkehrsteilnehmern zur Erhöhung der Sicherheit im Verkehr anbietet. Das AGEDA Projekt arbeitet hierzu mit MoveID und NoMo.bil eng zusammen.

Der Demonstrator besteht hier aus den folgenden fünf Komponenten, wie in Abbildung 5 dargestellt:

- **Das NeMo.bil Fahrzeug** erfasst die Fußgänger durch seine Kamera, die auf dem Zebrastreifen an dem Fahrzeug vorbeilaufen, mit Hilfe des ODS und überträgt sie an den TAS in der Cloud.
- **Das GAIA-X 4 AGEDA Fahrzeug** registriert sich über GAIA-X für den TAS und bezieht diese Informationen zu den erfassten Verkehrsteilnehmern, die helfen, Unfälle zu vermeiden.
- **Der GAIA-X 4 MoveID Authentication Service** stammt aus dem Schwesterprojekt MoveID und ermöglicht es dem Fahrzeug, sich für den TAS zu authentifizieren und zu registrieren.
- **Der GAIA-X 4 AGEDA TAS** empfängt die Objektinformationen des ODS über die Verkehrsteilnehmer vom NeMo.bil Fahrzeug und stellt sie anderen Fahrzeugen zur Verfügung, die für diesen Service registriert sind.
- **Der Carla Simulator** visualisiert die Vorgänge in der TAS Area, um die Abläufe dem Messebesucher visuell anschaulich zu vermitteln.



Abbildung 5: Übersicht der Demonstration und dem Zusammenspiel zwischen moveID, AGEDA und NeMo

Während der Implementierungsaktivitäten im Vorfeld und für die Demonstration auf der Messe selbst, hostete Bosch den Organization Credential Manager (OCM) des TAS in seiner Azure Cloud, der Serviceabonementanfragen in Echtzeit verarbeitete. Dies beinhaltete auch die Überprüfung der Gültigkeit der verwendeten Anmeldeinformationen durch den GAIA-X Compliance Service. Die Cloud Hosting Implementierung aus den IDunion Projekt, an dem Bosch ebenfalls beteiligt war, wurde für die Anforderungen an den Messedemonstrator angepasst.

Das durch Bosch bereitgestellte Azure Cloud Backend bestand aus einer Virtual Machine (VM) mit einer x86_64 Architektur, zwei vCPUs, 4 GB RAM und einem Betriebssystem basierend auf Ubuntu Server 22.04 LTS. Die Firewall wurde entsprechend konfiguriert, so dass der Remote Zugriff per SSH für Konfigurations-

und Updateaktionen sowie Serviceanfragen des TAS und des Credential Managers möglich waren. Sämtliche Konfigurations- und Updateaktivitäten konnten nur durch Bosch Mitarbeiter durchgeführt werden, da es nicht möglich war, Administrationsrechte auf die Projektpartner zu übertragen.

Ablauf der Demo

Unter der Regie von Bosch entstand eine User Story, die interessierten Messteilnehmern vorgeführt wurde. Der Messebesucher sitzt hierfür in einem virtuellen Fahrzeug und kann innerhalb der TAS-Zone, in die er einfährt, von den Informationen profitieren, die ihm das auf dem Messestand ausgestellte NeMo.bil Fahrzeug über diesen Service zusendet. Die TAS-Zone ist hierbei der Bereich, in dem sich die Sichtbehinderung befindet und die Informationen zu den nicht sichtbaren Objekten bereitgestellt werden können. Zur Visualisierung (siehe Abbildung 6) ist dieser Bereich durch ein rotes leuchtendes Band im Carla Simulator dargestellt.

Es wird für die Demonstration vorausgesetzt, dass der Nutzer des TAS sein „Verifiable Credential“, woraus er zur Authentifizierung seine „Verifiable Presentation“ erzeugt, bereits besitzt.

Ablauf der Demonstration am Messestand:

- Der TAS ist nicht installiert.
- Das System schlägt dem Fahrer vor, den TAS zu installieren, um so aktiv Unfälle vermeiden zu können.
- Der Fahrer entscheidet sich, den neuen Service zu installieren
(actor: Vehicle, message: You have successfully installed the Traffic Awareness Service)
- Das Fahrzeug fährt in die TAS-Zone
(actor: Vehicle, message: You are in an area where TAS is helping you! Do you want to use it now?)
- Der Fahrer entscheidet sich für die Verwendung
(actor: Driver, message: I want to use the Traffic Awareness Service)
- Der Service fordert die Vorlage der “Verifiable Presentation”
(actor: Authentication Service (MoveID), message: To use this service, you have to provide your Verifiable Presentation)
- Der Fahrer übermittelt die “Verifiable Presentation”, ohne dabei vom Verkehr abgelenkt zu werden
(actor: Driver, message: Ok, here is my Verifiable Presentation)
- Der Zugriff wird gewährt
(actor: Authentication Service (MoveID), message: I have successfully identified you!)
- Ein anderes Fahrzeug (NeMo.bil) an der Kreuzung beginnt damit, dem eigenen Fahrzeug Daten zuzusenden
(actor: NeMo.bil, message: I have detected pedestrians!)
- Das Fahrzeug zeigt Informationen an, die helfen, Kollisionen zu vermeiden
(actor: allVehicles, message: I am able to receive and share infos!)

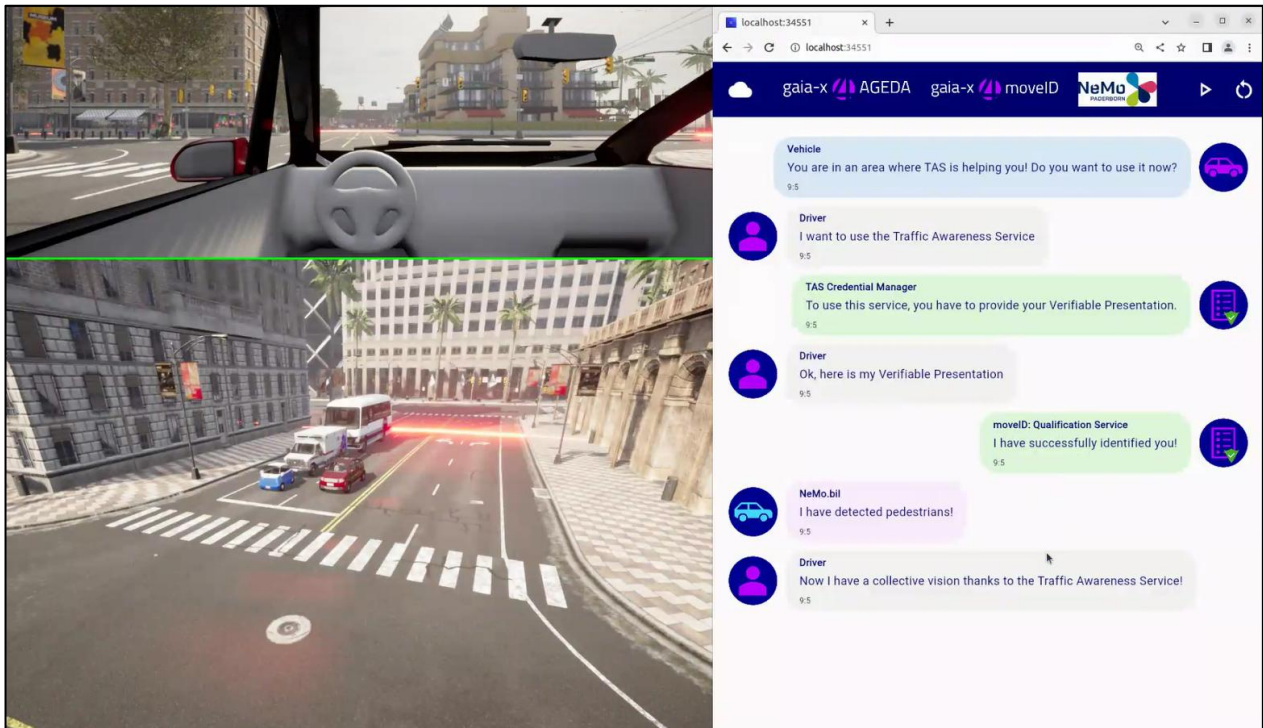


Abbildung 6: Ausschnitt aus der CARLA Simulation und dem Credential Interaction Flow

3.1.2.2 Versuchsfahrzeug

Für das Projekt AGEDA und die CVC-Level-4-Demonstration setzte die Robert Bosch GmbH einen VW Golf VII Variant als Versuchsträger ein, siehe Abbildung 7. Dieses Fahrzeug war bereits umfassend ausgestattet und kann Automatisierungsfunktionen verschiedener Stufen (SAE L1–L4) abbilden.



Abbildung 7: Eingesetztes Versuchsfahrzeug (VW Golf VII Variant) für die CVC-L4 Demonstration

Um das CVC-L4-Szenario zu realisieren, wurde das AGEDA-Framework in das Fahrzeug integriert und an die bestehende Entwicklungsumgebung angebunden. Dazu wurde die Hardware um einen Prototyping-Computer (Integration des AF) und einen Raspberry Pi erweitert, auf dem das AGEDA Framework installiert ist. Der Raspberry Pi ist über einen CAN-Adapter an eine CAN Breakout-Box angeschlossen und erhält somit Zugriff auf alle relevanten CAN-Signale im Fahrzeug. Des Weiteren ist ein GPS-Modul angeschlossen, um eine Lokalisierung zu ermöglichen. Über ein LTE-Surfstick erhält das AGEDA Framework eine Internetverbindung und kann Daten an den Traffic Awareness Cloud Service senden. Eine Übersicht des Fahrzeuges und seiner Komponenten ist in Abbildung 8 zu sehen.

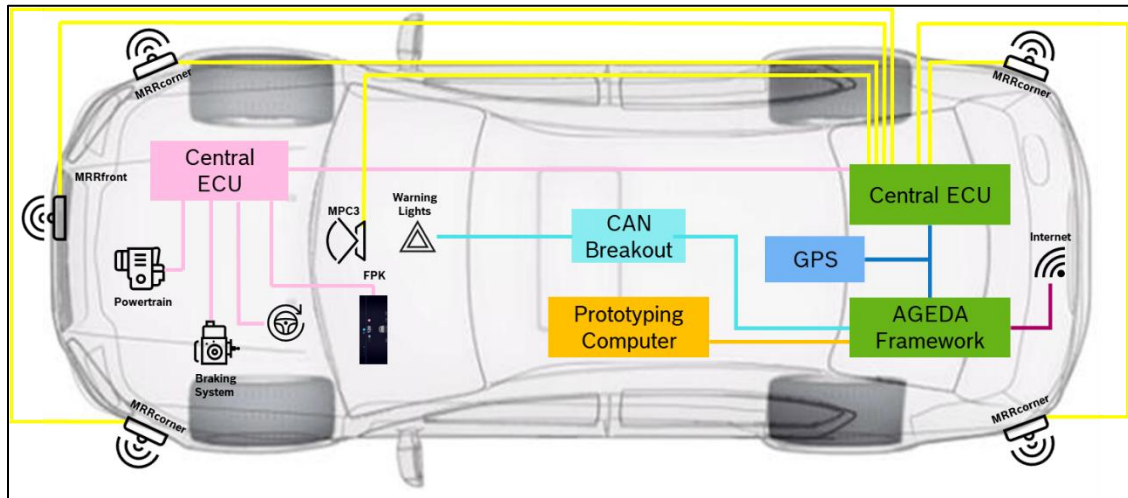


Abbildung 8: Vereinfachte Systemarchitektur des Fahrzeuges mit integriertem AGEDA Framework

Das Versuchsfahrzeug verfügt über ein modifiziertes Instrumentencluster das es ermöglicht, Inhalte frei und flexibel für den Fahrer darzustellen. Für den AGEDA-CVC-Level-4-Anwendungsfall wurde die bestehende Implementierung so erweitert, dass die Anzeige einer Gefahrenmeldung möglich ist. Dies ist in den nächsten zwei Abbildungen zu sehen.

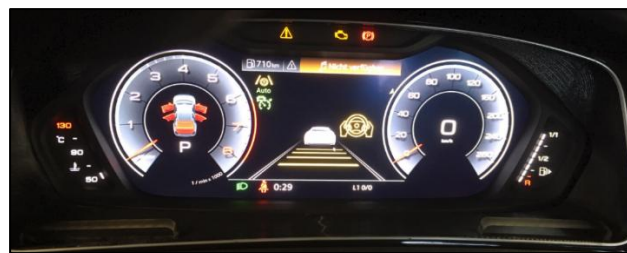


Abbildung 9: Modifiziertes Kombiinstrument mit der Möglichkeit, Inhalte frei zu bespielen

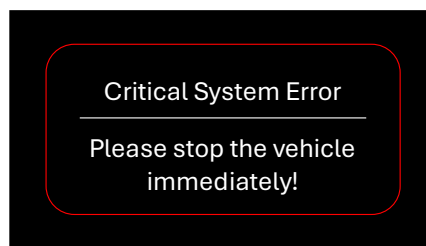


Abbildung 10: Implementierte Fehlermeldung zur realitätsnahen Umsetzung des CVC-L4 Szenarios

Für eine Beschreibung der entwickelten und verwendeten Software kann Abbildung 11 hinzugezogen werden. Dort sind die von Bosch entwickelten Softwarekomponenten in ihrer Beziehung zueinander dargestellt.

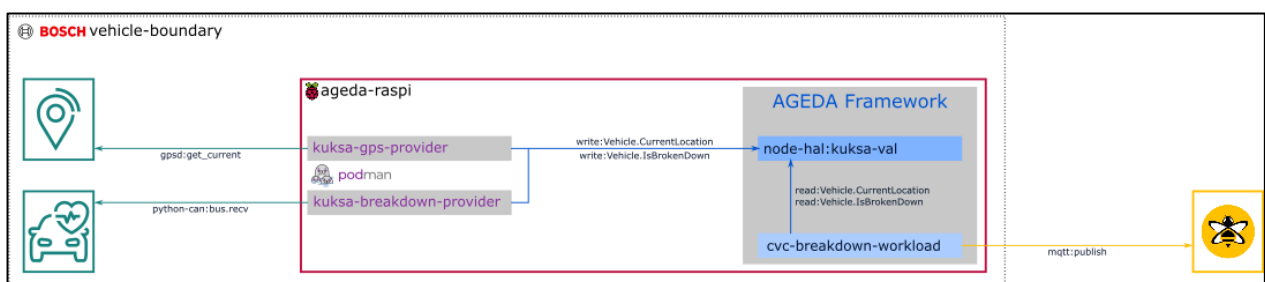


Abbildung 11: Bosch Softwarekomponenten - Demo 2025

Die entwickelten Komponenten decken zwei Bereiche ab. Zum einen wird Software benötigt, welche die Fahrzeuginformationen (GPS-Position und Status) für den Kuksa-Databroker bereitstellen. Auf der anderen Seite sorgt ein AGEDA-Workload dafür, dass diese Daten, nachdem sie vom Databroker abgeholt wurden, per Mobilfunknetz an den zentralen MQTT-Broker fertig formatiert übermittelt werden.

Der **kuksa-gps-provider** greift als containerisierte Applikation die aktuelle GPS-Position sowie die Fahrtrichtung ab und leitet diese unter Beachtung der VSS (vehicle-signal-specification) an den Kuksa-Databroker weiter. Eine weitere containerisierte Applikation, der **kuksa-breakdown-provider**, greift über den CAN-BUS die Information ab, ob das Warnblinklicht ein- oder ausgeschaltet ist. Diese Information wird dann genutzt, um den Wert für die Variable *isBrokenDown* im Databroker zu setzen.

Der AGEDA-Workload **cvc-breakdown-workload** ist über seine Verifiable-Credentials berechtigt, die GPS-Informationen des Fahrzeugs sowie die Variable *isBrokenDown* vom Kuksa Databroker abzufragen. Diese werden zyklisch abgefragt und in eine JSON-Nachricht verpackt, welche dann an den MQTT-Broker versendet wird.

Diese im Fahrzeug vereinten Bausteine bilden den Teil des Bosch-Fahrzeuges für die CVC-Demo und ermöglichen eine fusionierte Betrachtung der Verkehrslage unter Einbeziehung mehrerer Komponenten, wie Verkehrskameras, Fahrzeuge und Cloud Services.

Die Soft- und Hardware Architektur des Abschlussevents für das Bosch Fahrzeug lässt sich aus den vorigen Abschnitten ableiten und soll nochmal kurz beschrieben werden.

Hardwareseitig musste das Fahrzeug nicht umfassend angepasst werden. Es galt einen Rechner für das AGEDA Framework und die nachfolgenden Softwarekomponenten bereitzustellen. Dieser brauchte Zugriff auf GPS-Position und Warnblinklicht Status des Fahrzeuges. Zuletzt wurde auch noch eine Verbindung zum Internet benötigt.

Folgende Softwarekomponenten waren im Fahrzeug implementiert und lauffähig:

- Kuksa-gps-provider
- Kuksa-breakdown-provider
- AGEDA Framework:
 - Gaia-X-Connector
 - Platform-Services
 - Edge-Enabler
 - SOVD-Server Plugin
 - Kuksa-Databroker Plugin
 - CVC-Breakdown Workload

Cloudseitig wurden folgende Softwarekomponenten benötigt:

- SOVD Client Webansicht
- MQTT-Broker für den Nachrichtenaustausch
- Birds-Eye-GUI
- Image-Artifactory
- VC-Katalog

Bei der Konzeption und Umsetzung wurden sowohl die Anforderungen des AGEDA-Frameworks berücksichtigt als auch spezifische Anforderungen an das AGEDA-Framework definiert, um eine optimale Integration des SOVD-Servers sicherzustellen. Die Entscheidung über die Realisierungsvariante – ob als AGEDA Building Block oder als Workload – erfolgte in enger Abstimmung mit dem entsprechenden Arbeitspaket, um technische und organisatorische Abhängigkeiten zu minimieren und eine konsistente Architektur zu gewährleisten, vgl. Kapitel 3.3.1.2.3.

3.1.2.3 Monitoring @ CVC L3 Demo (WP1.3)

Für die CVC L3 Demo wurde eine erste Implementierung des Diagnose- und Monitoringkonzepts auf SOVD-Basis im Rahmen von AGEDA sowohl auf der AGEDA-ECU für das DLR-Fahrzeug als auch für den Table-Demonstrator, der beim Konsortialtreffen in Wolfsburg am 15.11.2024 vorgestellt wurde, erfolgreich von Bosch und AVL DiTEST durchgeführt. Abbildung 12 zeigt die Komponenten, die für das Setup benötigt werden:

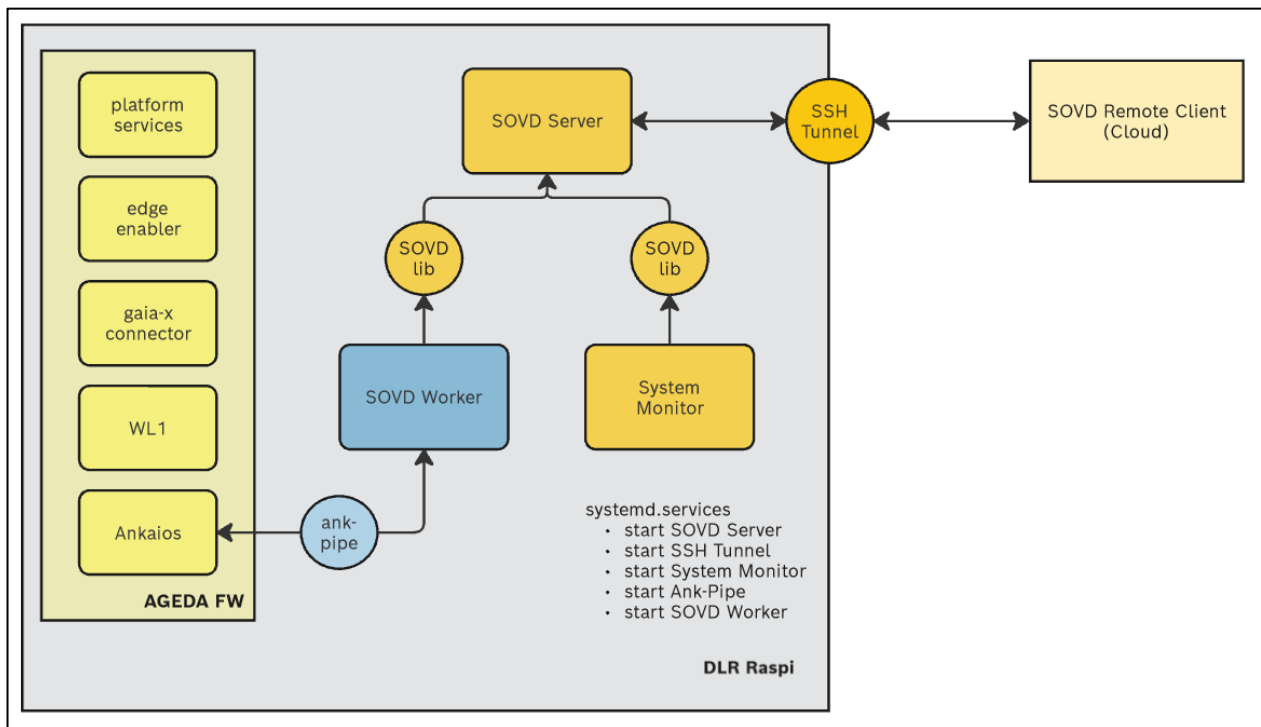


Abbildung 12: SOVD-Demo 2024 Setup

In diesem ersten Setup wird eine Instanz des SOVD-Servers auf einem Raspberry Pi installiert. Als Datenquellen dienen zum einen der SOVD-Worker, dieser liefert Informationen zum AGEDA Framework, und ein System Monitor Service für allgemeine Systeminformationen. Beide sind mit der SOVD-Programmbibliothek (libsovd) ausgestattet, welche die Anbindung an den SOVD-Server ermöglicht. Die Nachrichten sind mittels Protobuf serialisiert.

Name	Id	Link
Health Monitoring	HealthMonitoring	http://localhost:3000/apps/HealthMonitoring
Gaia-X App 1	gaia-x-app-1	http://localhost:3000/apps/gaia-x-app-1
Gaia-X App 2	gaia-x-app-2	http://localhost:3000/apps/gaia-x-app-2
AnkaiosWorkloads	AnkaiosWorkloads	http://localhost:3000/apps/AnkaiosWorkloads

Abbildung 13: Browser-Benutzeroberfläche des AVL DiTEST Remote-Clients-Web-Client

Die Verbindung zum SOVD-Remote Client in der Cloud wurde mittels reverse SSH-Tunnel realisiert, der über einen weiteren AVL DiTEST-Server mit Firewall aufgebaut wurde. Auf dem Remote-Client lassen sich Diagnoseinformationen anzeigen. Abbildung 13 zeigt die Benutzeroberfläche des Remote-Clients, welcher eine gezielte Untersuchung der Daten vereinfacht.

Nach dieser ersten Implementierung des SOVD-Servers soll die nächste Implementierung näher an das AGEDA Framework rücken. Hierfür werden im nächsten Abschnitt drei Varianten beleuchtet.

3.1.2.4 Abschlussevent | Fahrzeug- & Table-Demonstrator

Am 22. Oktober fand an der Hochschule Hamm-Lippstadt das Abschlussevent des Forschungsprojekts AGEDA statt. Das Team präsentierte erfolgreich die Funktionalitäten und Ergebnisse der Integration des AGEDA Frameworks in ein Versuchsfahrzeug der Robert Bosch GmbH, insbesondere im Kontext von Monitoring und Update. Das Framework wurde an die bestehende Fahrzeugarchitektur angeschlossen und mit Zugriff auf relevante Fahrzeugsignale ausgestattet. Die gewonnenen Erkenntnisse und Funktionen wurden sowohl an einem Table Demonstrator als auch direkt am Versuchsfahrzeug der Robert Bosch GmbH vorgestellt.

Table Demonstrator: From Code to Road – Utilizing the AGEDA Framework

Der finale Table-Demonstrator dient dazu, die Funktionalität des AGEDA-Frameworks über alle Phasen des Lebenszyklus eines Services darzustellen – von der Entwicklung über die Verifikation und den Rollout in die Fahrzeugflotte bis hin zum Betrieb auf der Straße. Dabei wird die Interaktion mit allen beteiligten Akteuren, einschließlich OEM, Service-Provider und Endnutzer, nachvollziehbar gemacht, siehe Abbildung 14.

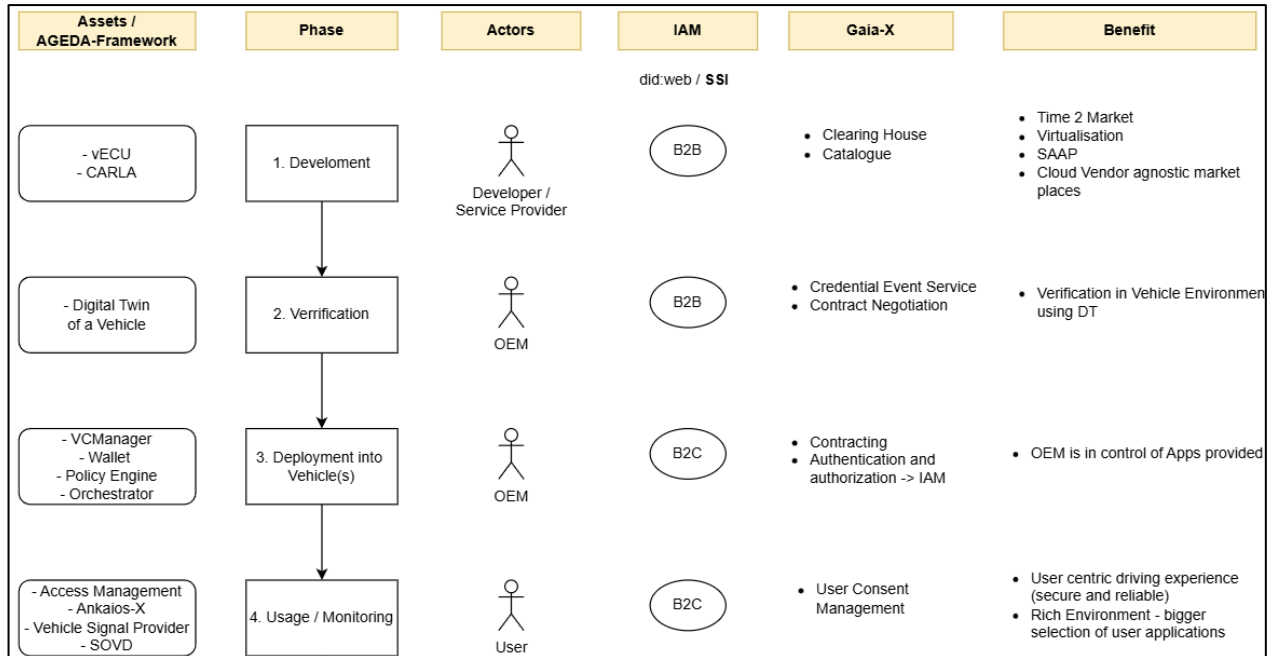


Abbildung 14: Workflow und Bestandteile des Table Demonstrators

Als exemplarischer Anwendungsfall wird ein **Collision-Warning-Service** verwendet. Dieser Service wird vom OEM initiiert, von einem Service-Provider entwickelt und schließlich vom Fahrer genutzt. Die Funktion basiert auf der Verarbeitung vorhandener Fahrzeugsensordaten, um den Fahrer vor potenziellen Kollisionen mit Verkehrsteilnehmern zu warnen.

Phase 4 - Betriebsphase und Monitoring

In Phase 4, dem Betrieb, kommt ein umfassendes Monitoring zum Einsatz. Hierfür wird eine Remote-Diagnoselösung auf Basis des **Service-Oriented Vehicle Diagnostic (SOVD)**-Standards genutzt. Der Collision-Warning-Service ist über die SOVD-Library an den SOVD-Server angebunden. Dadurch können sowohl der OEM als auch der Service-Provider relevante Informationen zur Interaktion des Services mit dem Fahrzeug sowie mit dem Backend einsehen. Dies ermöglicht eine kontinuierliche Überwachung und Wartung des AGEDA-Frameworks und seiner Services, siehe Abbildung 15 und Abbildung 16.

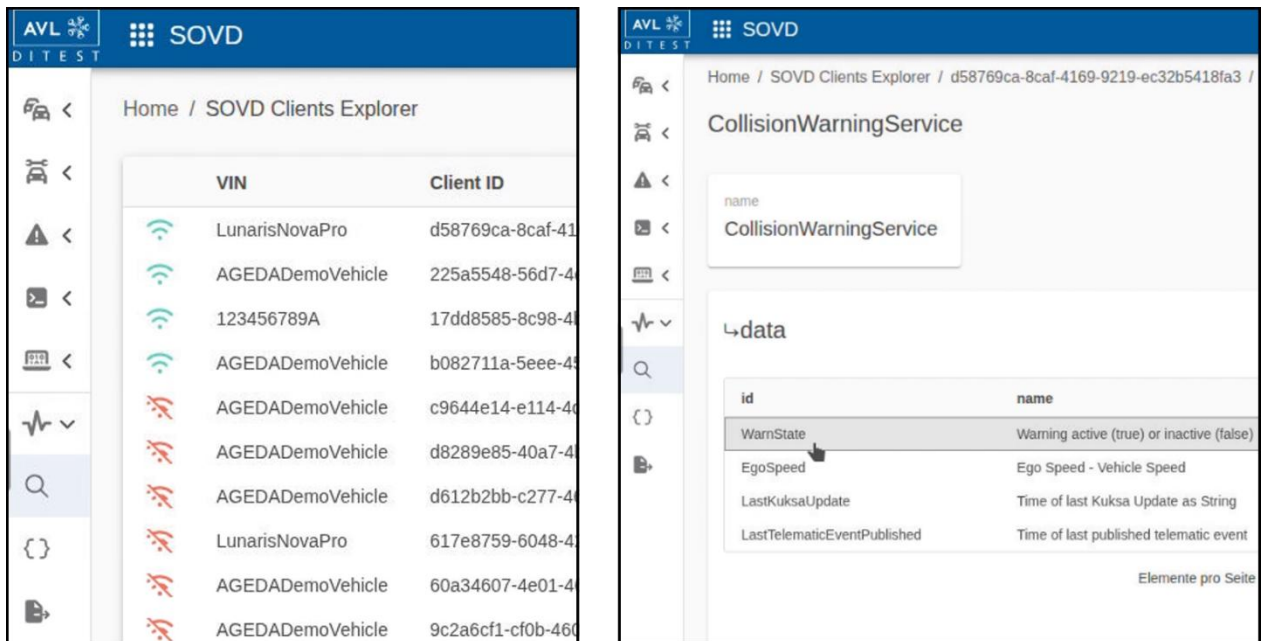


Abbildung 15: Remote SOVD-Client vom Table Demonstrator

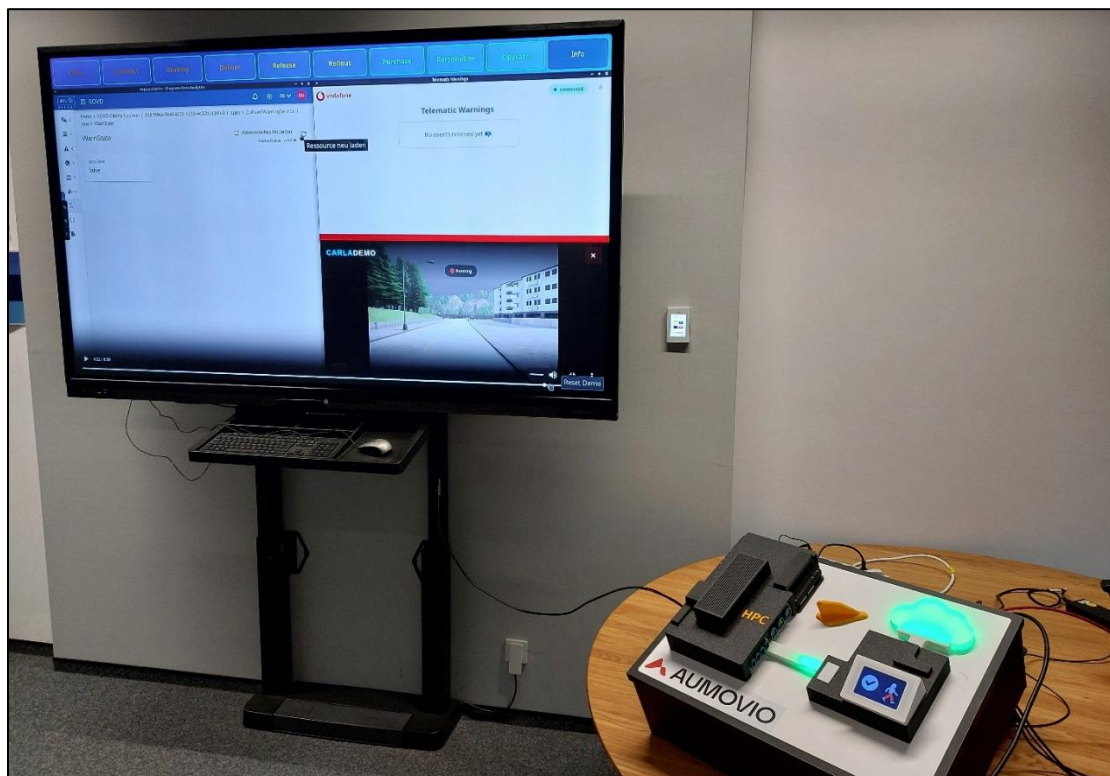


Abbildung 16: Foto vom Table Demonstrator vom Abschlussevent in Lippstadt Nov. 2025

Fahrzeugdemonstration Update & Monitoring

Im Zentrum der Präsentation am Versuchsfahrzeug der Robert Bosch GmbH standen die Monitoring- und Update-Use Cases. Hierfür wurde der SOVD-Server als zentraler Building Block im AGEDA Framework implementiert und die relevanten Workloads mit einem entsprechenden Interface versehen, sodass sie unter anderem auch Zugriff auf Fahrzeuginterne Signale hatten. Die Präsentation erfolgte direkt am Versuchsfahrzeug, wobei eine Kombination aus Tablets und Postern zum Einsatz kam (siehe Abbildung 19).

Ein dedizierter Monitor visualisierte den Traffic Awareness Service mit seinen aktuellen Statusinformationen, während über ein Tablet die detaillierten Informationen über den SOVD-Client im Backend abgerufen werden konnten. Ein zusätzliches Poster lieferte Hintergrundinformationen zur Architektur des Systems.

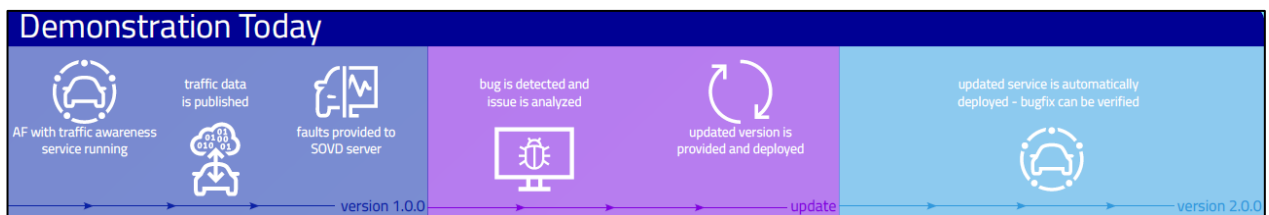


Abbildung 17: Ablauf der Demo

Ablauf der Demonstration:

Die Demo durchlief drei wesentliche Phasen (siehe Abbildung 17), die die Leistungsfähigkeit des AGEDA Frameworks im Kontext Monitoring & Update eindrucksvoll unter Beweis stellten:

- 1. Initialisierung und Monitoring:** Zu Beginn waren das AGEDA Framework und der "Breakdown Workload" gestartet. Aktuelle Fahrzeugdaten sowie Informationen des Workloads konnten über den SOVD-Client im Backend abgefragt und analysiert werden. Parallel dazu visualisierte eine GUI-Oberfläche den Traffic Awareness Service.
- 2. Automatisierte Fehlererkennung und -behebung:** Die Simulation eines Fehlers erfolgte durch Trennen der Verbindung zum GPS-Empfänger. Der "Breakdown Workload" erkannte diesen Fehler umgehend und kommunizierte ihn an den SOVD-Server, der die Information wiederum dem Backend zur Verfügung stellte. Nach der Wiederherstellung der GPS-Verbindung heilte sich der Fehler automatisch selbst, und der entsprechende Fehlereintrag wurde gelöscht.
- 3. Nahtloses Software-Update:** Abschließend wurde ein realistisches Szenario eines Softwarefehlers simuliert: Das Heading des Fahrzeugs wurde im Stillstand nicht korrekt angezeigt. Das Entwicklungsteam stellte daraufhin eine neue Version (V2.0.0) im "Minimal Catalogue" bereit. Die Policy Engine des AGEDA Frameworks glied die Versionen ab, stoppte den laufenden Workload, führte das Update durch und startete anschließend die neue, korrigierte Version des Workloads. Dieser Prozess verdeutlichte die effiziente und automatisierte Möglichkeit zur Software-Aktualisierung im Feld.

Die Präsentation demonstrierte eindrucksvoll das Potenzial des AGEDA Frameworks, komplexe Fahrzeugsysteme effizient zu überwachen, Fehler automatisch zu erkennen und zu beheben sowie Software-Updates nahtlos zu orchestrieren.

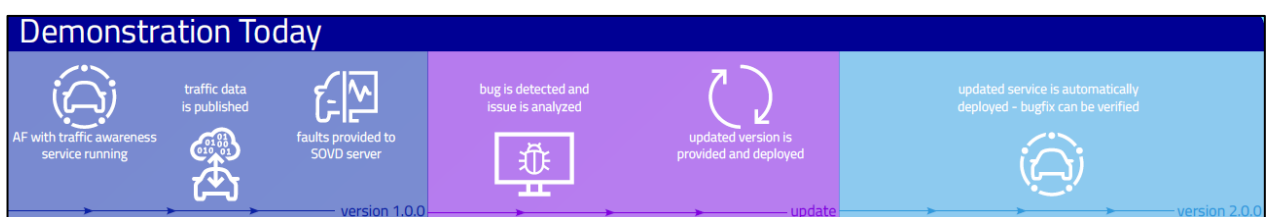


Abbildung 18: Ablauf der Demo



Abbildung 19: Präsentation der finalen Projektergebnisse von AVL DiTest und Robert Bosch GmbH zum Thema Monitoring und Update am Versuchsfahrzeug der Robert Bosch GmbH.

3.1.3 AP 1.4: Verifikation und Validierung

Im Rahmen des AP 1.4 hat Bosch aktiv an den Abstimmungsterminen zu den Themen Verifizierung und Validierung nach dem V-Modell teilgenommen und die vorhandene Expertise in die Erstellung eines Testkonzepts eingebracht, ohne dabei das Forschungsprojekt durch strenge Validierungsanforderungen und -prozesse zu gefährden. Der Fokus lag auf der Definition und Durchführung der notwendigen Validierungsmaßnahmen für die in Kapitel 3.3.1.2 beschriebene Monitoring-Funktion. Die Arbeitsergebnisse wurden im Confluence dokumentiert.

3.2 Teilprojekt 2: Bruchlose Integration von GAIA-X Gestaltungsprinzipien in innovative Fahrzeugarchitekturen

3.2.1 AP 2.1: Software- und Hardware-Architektur und AGEDA Komponenten

Im Arbeitspaket 2.1 beteiligte sich BOSCH an der kontinuierlichen Verfeinerung und Weiterentwicklung der AGEDA Hauptkomponenten, sowie der Plugin-Architektur. Als Entwickler von Workloads sowie Plugins im Bereich Monitoring oder als Teil der Demo wurde durch das Stellen von Anforderungen und Öffnen von git-issues die Architektur des AGEDA Framework aktiv mitgestaltet.

Als exemplarischer Service-Provider hat Bosch zusätzlich einen minimal-catalog mitentwickelt und gepflegt, welcher als erste Informationsquelle für gewünschte Fahrzeugservices dient und auch genutzt wird. Die weitere Entwicklung und Platzierung eines solchen Kataloges sind noch zu klären.

3.2.1.1 GAIA-X Architecture

In der zweiten Projektzeithälfte beschäftigte sich Bosch intensiver mit dem GAIA-X Themenkomplex. Der Fokus lag dabei auf den digitalen Identitäten im Fahrzeugkontext. Nach einer ausführlicheren Einarbeitung beteiligte sich Bosch an der Arbeitsgruppe und übernahm die stellvertretende Leitung hinter Vodafone. Für den Meilenstein 36 wurden gemeinsame Ziele definiert und Verantwortlichkeiten festgelegt. Nachfolgend sind die Ziele mit Bosch Beteiligung aufgeführt:

Tabelle 2: Übersicht über die GAIA-X Ziele für den M36 mit Bosch Beteiligung ([Auszug aus Confluence](#))

Goal 07	Digital Identities and VC/VP for different User Journeys	Leitung
Goal 08	Schema extensions for VCs	Mitarbeit
Goal 10	E2E GX architecture	Mitarbeit

Im Rahmen von Ziel 7 bestand die Aufgabe darin, verschiedene Anwenderszenarien (User Journeys) für die Nutzung digitaler Identitäten und digitaler Nachweise (Verifiable Credentials, VCs) zu entwickeln. Die ursprüngliche User Story lautete:

As an architect, I want to have a better understanding about the usage of Digital Identities (SSI stacks) and VC / VP for different User Journeys in the vehicle context so that user journeys can be implemented.

In Zusammenarbeit mit ITK-Engineering und Continental wurden entsprechende User-Journeys mit Verifiable Credential (VC) Austausch für einen Personalisierungs Use Case und einen Zugangsberechtigungs- Use Case entwickelt und diskutiert. Ein weiterer Fokus vom Arbeitspaket lag auf der Interaktion zwischen fahrzeug- und personenbezogenen Nachweisen.

Basis für das Sequenz Diagramm waren die Ergebnisse aus Ziel 8. Die Arbeitsgruppe beschäftigte sich mit den Schemata, die VCs für das AGEDA Framework definieren. Schnittstellen, an denen VCs verwendet werden, wurden identifiziert. Dazu mussten die Rollen verstanden werden, die innerhalb von AGEDA unter Verwendung von VCs interagieren. AGEDA fungiert dabei als Halter von Gaia-X konformen VCs und als Aussteller benutzerspezifischer VCs. In Abbildung 20 sind die Rollen und die Interaktionen dargestellt. Für die Erstellung der anschließenden Sequenzdiagramme ist wichtig, dass zwischen OEM und Service Provider eine B2B-Beziehung und zwischen OEM und End User eine B2C-Beziehung besteht. Es entsteht keine Geschäftsbeziehung zwischen End User und Service Provider. Das entstehende Vertrauensmodell ist OEM-zentrisch.

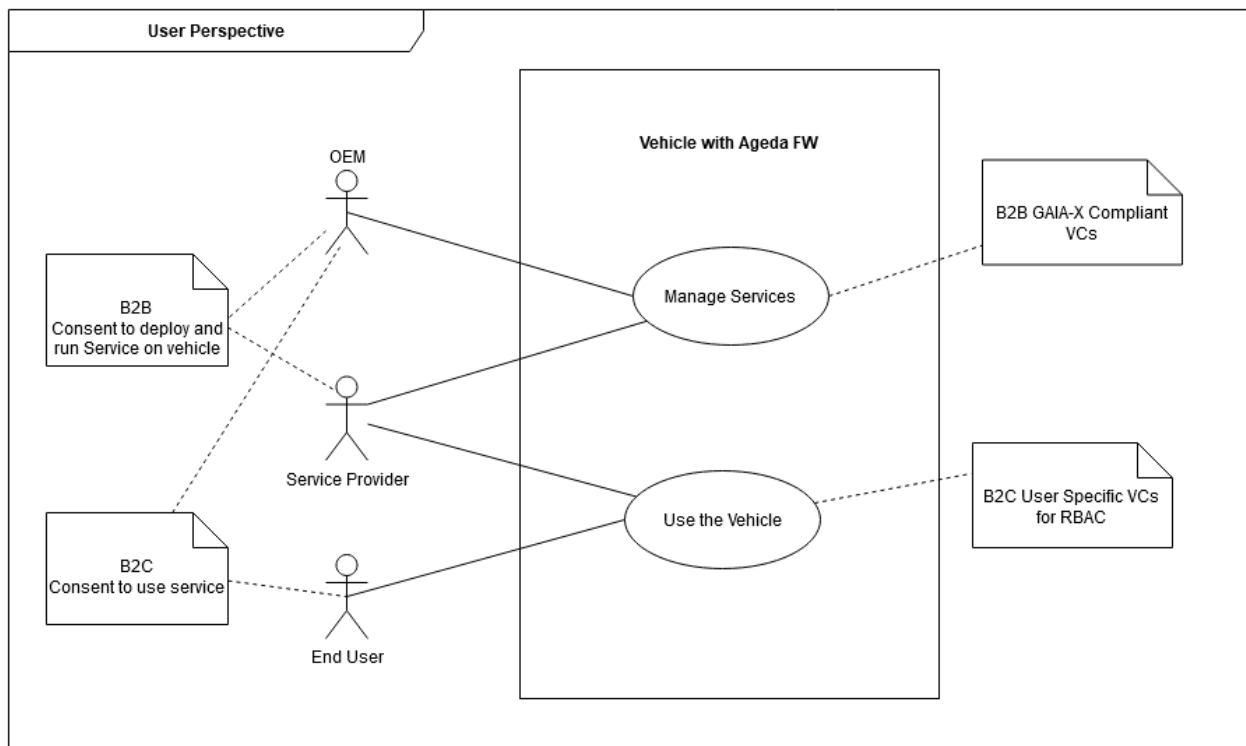


Abbildung 20: Übersicht über die Geschäftsbeziehungen der involvierten Partner

User Journey Personalisierung:

Im Kontext Personalisierung wurden zwei unterschiedliche Fokusthemen, mit dem personenabhängigen Start von Services und der allgemeinen Personalisierung von Fahrzeugsystemen, identifiziert.

Bei der Nutzung von Services muss im Vorfeld ein Vertrag - zumindest in Form der Akzeptierung von Nutzungsbedingungen - zustande kommen. Im Sinne von Gaia-X wird dieser Vertrag von einer nutzenden Person eingegangen. In Zukunft kann der Fall auftreten, dass in demselben Fahrzeug Nutzer A einen Service abonniert, Nutzer B jedoch nicht. Folglich darf der Service nur für Nutzer A gestartet werden. Dieses Szenario hat große Datenschutzrelevanz, wenn im Rahmen des Services z. B. Bewegungsdaten vom Nutzenden geteilt werden. Zukünftig muss dementsprechend gewährleistet werden, dass ausschließlich Services gestartet werden, welche der aktuell Nutzende zugestimmt hat.

Das zweite Szenario umfasst die allgemeine Personalisierung von Fahrzeugeinstellungen wie Infotainment, Fahrmodus oder Fahrwerk. Dies ist bereits heute schon über verschiedene Mechanismen möglich, jedoch bietet Gaia-X neue Möglichkeiten und Schutzmechanismen für personenbezogene und sensible Daten.

Das Sequenzdiagramm in Abbildung 21 visualisiert den Ablauf und den Fluss der Verifiable Credentials. Der Ablauf kann in vier Phasen unterteilt werden, die nachfolgend beschrieben werden:

Phase 1: Dienstfreigabe (Release Service)

Ein Serviceanbieter stellt einen neuen Service in einem Katalog bereit. Der Fahrzeughersteller (OEM) durchsucht den Katalog und wählt einen Service aus. Anschließend verhandeln OEM und Anbieter die Konditionen, tauschen ihre digitalen Nachweise aus und verifizieren diese gegenseitig. Der OEM speichert den erhaltenen Nachweis in seiner DID-Wallet.

Der OEM legt damit fest, welchen Anbietern er vertraut und welche Dienste für seine Fahrzeugflotte freigegeben werden. Zwischen OEM und Anbieter wird ein Vertrag geschlossen.

Phase 2: Bereitstellung des Services (Download Service)

Das Fahrzeug lädt alle vom OEM freigegebenen Services herunter. Dieser Schritt erfolgt unabhängig davon, ob ein Endnutzer die Dienste bereits abonniert hat.

Phase 3: Abonnement durch den Endnutzer (End User Subscription)

Der Endnutzer sieht in einem fahrzeuginternen Katalog ausschließlich die vom OEM vorausgewählten und zugelassenen Services. Wählt er einen Service aus, findet ein Austausch von Nachweisen statt und der Nutzer speichert den resultierenden Nachweis in seiner persönlichen Wallet. Dadurch wird ein Vertrag zwischen dem OEM und dem Endnutzer abgeschlossen. Der OEM fungiert hierbei als Vermittler und es entsteht keine direkte Vertragsbeziehung zwischen Nutzer und Dienstanbieter.

Phase 4: Start des Services im Fahrzeug (Start of Service)

Wenn der Nutzer das Fahrzeug betritt und sich damit verbindet (z. B. mit Hilfe seines Smartphones), präsentiert er seine digitalen Nachweise. Ein Agent der fahrzeugeigenen DID-Wallet verifiziert diese. Bei erfolgreicher Prüfung generiert das AGEDA-Framework die erforderlichen Sicherheitstoken (z. B. für Kuksa.val), woraufhin die entsprechenden Services gestartet werden. Es werden ausschließlich jene Services aktiviert, für die der Nutzer einen gültigen digitalen Nachweis besitzt.

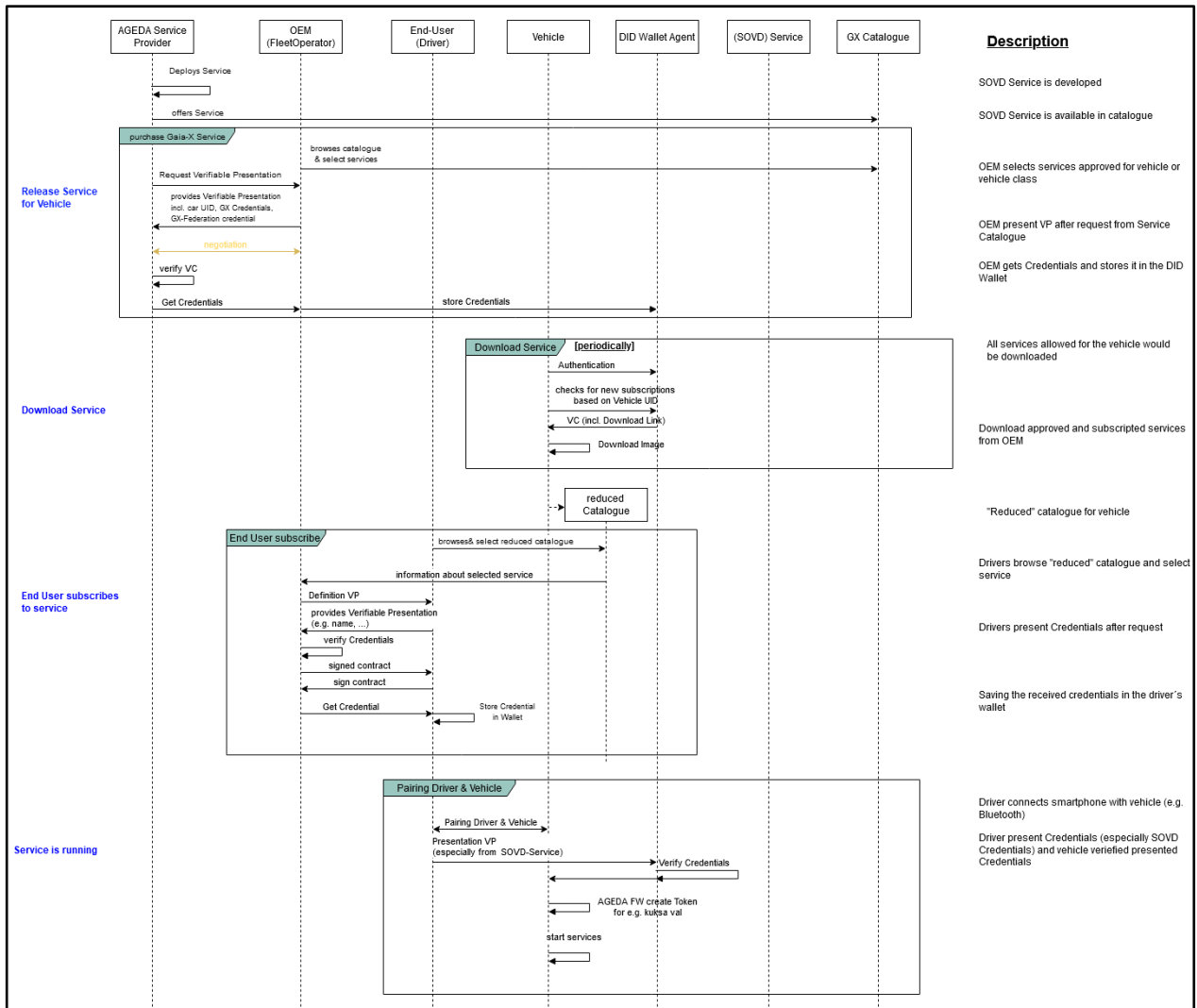


Abbildung 21: Sequenz Diagramm Zugangsberechtigung Use Case

User Journey Zugangsberechtigung:

Ein weiterer interessanter Anwendungsfall ist die Gewährung von Zugang zu zugangsbeschränkten Bereichen durch den Austausch digitaler Nachweise. Ziel ist es, dem Fahrzeug zu ermöglichen, den Zugang zu einem Parkhaus oder - allgemeiner - zu einem Sperrbereich zu verwalten. Das bedeutet, dass keine Benutzerinteraktion am Parkhaus mehr notwendig ist. Darüber hinaus kann das Fahrzeug auch die Bezahlung verwalten.

Der Anwendungsfall basiert weitgehend auf dem Personalisierungs-Use-Case, nur mit einer kleinen Erweiterung am Ende.

Phase 1: Dienstfreigabe (Release Service)

Analog zum Personalisierungs-Use-Case

Phase 2: Bereitstellung des Services (Download Service)

Analog zum Personalisierungs-Use-Case

Phase 3: Abonnement durch den Endnutzer (End User Subscription)

Analog zum Personalisierungs-Use-Case

Phase 4: Start des Services im Fahrzeug (Start of Service)

Im nächsten Schritt steigt der Benutzer in das Fahrzeug und koppelt sich mit diesem. Der Endbenutzer präsentiert die Nachweise, das Fahrzeug - insbesondere der DID Wallet Agent - überprüft diese und erstellt dann über das AGEDA Framework die notwendigen Tokens (z. B. für Kuksa.val) und startet den Dienst. Das bedeutet, dass nur Services gestartet werden, für die der Endbenutzer Nachweise besitzt.

Zusätzlich zum Personalisierungs-Use-Case erstellt das AGEDA Framework auch Tokens für den Parkservice und stellt diese Tokens dem Service zur Verfügung.

Phase 5: Gewährung des Zugangs

Im letzten Schritt nähert sich das Fahrzeug dem Parkhaus. Eine Verbindung wird hergestellt. Das Fahrzeug präsentiert die Token und die Information über die unmittelbar bevorstehende Schranke. Zusätzlich kann auch die Schranke Informationen über das Fahrzeug bereitstellen bzw. verifizieren. Der Cloud-Dienst überprüft alle Informationen und sendet den Auslöser zum Öffnen der Schranke.

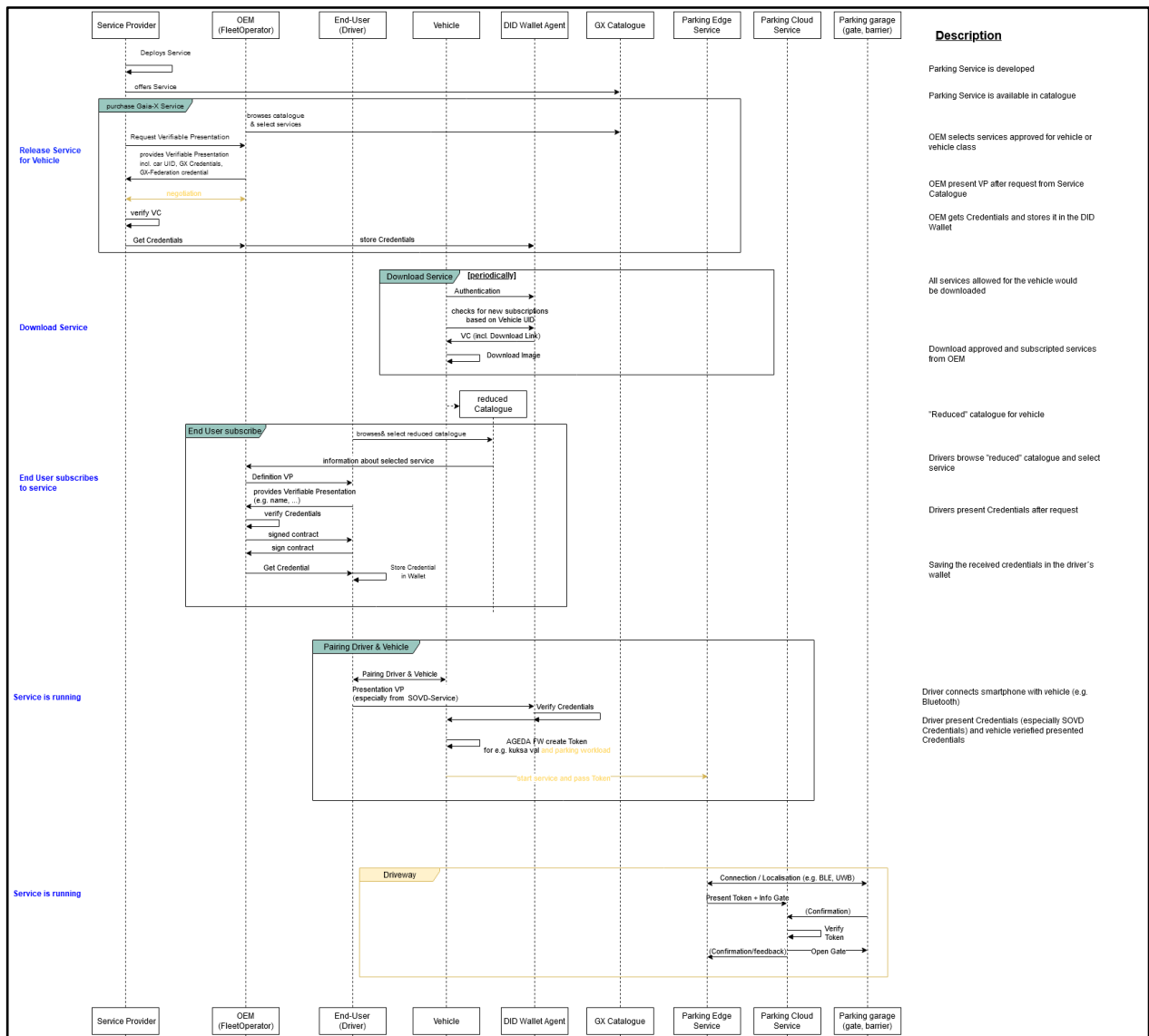


Abbildung 22: Sequenzdiagramm Zugangsberechtigung Use Case

3.2.2 AP 2.3: Dynamische Rekonfiguration

Bosch hat an regelmäßigen Abstimmungsterminen des Arbeitspakets teilgenommen. Entsprechende Rekonfigurationsszenarien basierend auf den Update & Monitoring Use-Cases wurden definiert und ein Performance Degradierungsplan wurde erarbeitet.

3.3 Teilprojekt 3: Methoden, Werkzeuge und Zertifizierung

3.3.1 AP 3.2: Betrieb

Das Arbeitspaket 3.2 entwickelt Methoden und Werkzeuge, um den Betrieb von Cloud-/Edge-basierten Fahrzeugfunktionen kontinuierlich zu überwachen und gezielt Updates, Upgrades sowie Rekonfigurationen sicher und effizient ins Feld zu bringen. Es ist entscheidend für die Zuverlässigkeit und Weiterentwicklung vernetzter Fahrzeugfunktionen. Mit zunehmender Komplexität und Variantenvielfalt reicht die Laborvalidierung nicht mehr aus; ein kontinuierliches Monitoring im Feld wird notwendig, um:

Funktionsqualität und Sicherheit auch nach Auslieferung sicherzustellen. Agile Weiterentwicklung durch schnelle Rückkopplung in den Entwicklungsprozess zu ermöglichen. Effiziente Update-Prozesse zu etablieren, die sowohl technische Herausforderungen (Cloud-Edge-Interoperabilität, Datenvolumen) als auch regulatorische Anforderungen erfüllen. Damit bildet AP 3.2 die Grundlage für einen lebenszyklusorientierten Ansatz, der Fahrzeugsoftware dynamisch und sicher an neue Anforderungen anpasst.

Die nachfolgenden Kapitel unterteilen sich in Ausgangslage und Motivation im Kontext Monitoring und Update, sowie einer anschließenden Detailbetrachtung zu Monitoring und Update.

3.3.1.1 Ausgangslage und Motivation

Der Bereich Update und Monitoring in Fahrzeugen erlebt durch das Paradigma des **Software-Defined Vehicle (SDV)** eine revolutionäre Transformation. SDV bezeichnet eine Architektur, bei der die Funktionalität eines Fahrzeugs maßgeblich durch Software definiert und gesteuert wird, anstatt durch fest verdrahtete Hardware. Dies ermöglicht eine bisher unerreichte Flexibilität, Personalisierung und die Möglichkeit, neue Funktionen und Verbesserungen über den gesamten Lebenszyklus des Fahrzeugs bereitzustellen.

Software-Defined Vehicle (SDV) – Der Paradigmenwechsel:

- **Abstraktion von Hardware:** Software wird von spezifischer Hardware entkoppelt, wodurch Funktionen unabhängig von der zugrunde liegenden Plattform entwickelt und ausgerollt werden können.
- **Zentralisierung der Architektur:** Weg von verteilten ECUs hin zu zonalen Architekturen oder gar einem zentralen High-Performance-Computer, der die Software hostet.
- **Over-the-Air (OTA) Updates:** Schlüsseltechnologie, die es ermöglicht, Software-Updates und neue Funktionen drahtlos an das Fahrzeug zu übertragen, ohne einen Werkstattbesuch. Dies ist die Grundlage für kontinuierliche Verbesserung und Feature-on-Demand-Dienste.
- **Funktionsentwicklung als Software-Produkt:** Fahrzeugfunktionen werden als modulare, aktualisierbare Softwarepakete betrachtet, die von einem Ökosystem von Entwicklern beigesteuert werden können.

Monitoring im SDV-Kontext:

Das Monitoring in SDVs ist komplexer und gleichzeitig leistungsfähiger als in traditionellen Fahrzeugen. Es geht über die reine Fehlerdiagnose hinaus und umfasst:

- **Umfassende Datenakquisition:** Erfassung einer riesigen Menge an Daten von Sensoren, Aktuatoren, Steuergeräten und der Software selbst. Dies umfasst Betriebsdaten, Leistungsmetriken, Diagnosedaten (DTCs), Umweltinformationen und Nutzungsverhalten.
- **Echtzeit-Telemetrie:** Übertragung relevanter Daten an Backend-Systeme in Echtzeit oder nahezu Echtzeit zur sofortigen Analyse und Reaktion. Dies ist entscheidend für vorausschauende Wartung, die Erkennung kritischer Ereignisse und die Überwachung der Fahrzeugsicherheit.
- **Predictive Maintenance (Vorausschauende Wartung):** Einsatz von KI/ML-Modellen, um Muster in den Monitoring-Daten zu erkennen, die auf bevorstehende Ausfälle oder Wartungsbedarfe hinweisen. Ziel ist es, Probleme zu beheben, bevor sie auftreten.
- **Performance Monitoring:** Überwachung der Leistung von Software-Workloads, CPU-Auslastung, Speichernutzung und Netzwerkverkehr, um Engpässe zu identifizieren und die optimale Funktionalität sicherzustellen.
- **Security Monitoring:** Kontinuierliche Überwachung auf ungewöhnliche Aktivitäten oder potenzielle Cyberangriffe auf die Fahrzeugsysteme.
- **Health Monitoring:** Überwachung des allgemeinen Zustands des Fahrzeugs und seiner Komponenten auf Basis von Software- und Hardwaredaten.
- **Standardisierung (z.B. SOVD, VSS):** Es gibt Bestrebungen, die Übertragung und den Austausch von Fahrzeugdaten zu standardisieren (z.B. Service-Oriented Vehicle Diagnostics - SOVD, Vehicle Signal Specification - VSS), um Interoperabilität und eine breitere Nutzung der Daten zu ermöglichen.

Update-Mechanismen im SDV-Kontext:

Software-Updates sind das Herzstück des SDV-Konzepts und erfordern robuste, sichere und flexible Mechanismen:

- **Over-the-Air (OTA) Updates:**
 - **Full Vehicle Updates:** Aktualisierung des gesamten Software-Stacks des Fahrzeugs.
 - **Delta-Updates:** Übertragung nur der geänderten Teile der Software, um Bandbreite zu sparen und die Update-Zeiten zu verkürzen.
 - **Micro-Updates:** Aktualisierung einzelner Workloads oder Funktionen ohne Beeinträchtigung des Gesamtsystems.
- **Rollback-Fähigkeit:** Essenziell für die Sicherheit. Im Falle eines Fehlers während oder nach einem Update muss das System in der Lage sein, auf eine vorherige, stabile Softwareversion zurückzuspringen.
- **Sichere Update-Prozesse:**
 - **Authentifizierung und Autorisierung:** Sicherstellung, dass nur autorisierte Updates von vertrauenswürdigen Quellen installiert werden können.
 - **End-to-End-Verschlüsselung:** Schutz der Update-Pakete während der Übertragung und Speicherung.
 - **Verifikation und Validierung:** Überprüfung der Integrität und Authentizität der Update-Pakete vor der Installation.
 - **Fehlerbehandlung:** Robuste Mechanismen zur Behandlung von Stromausfällen, Verbindungsabbrüchen oder anderen Störungen während des Updates.
- **Dependencies und Orchestrierung:** Das Management von Updates in einem komplexen System erfordert die Berücksichtigung von Abhängigkeiten zwischen verschiedenen Softwarekomponenten und deren Orchestrierung (z.B. welche Komponente muss zuerst aktualisiert werden, welche Workloads müssen gestoppt werden).
- **Policy-Engine und Update-Strategien:** Einsatz von Regelwerken (Policies), die definieren, wann, wie und unter welchen Bedingungen Updates durchgeführt werden (z.B. nur im Stillstand, nur bei ausreichendem Batteriestand, nur in bestimmten Regionen).

- **Update-Feedback und Reporting:** Überwachung des Erfolgs von Updates und Bereitstellung von Feedback an das Backend-System.

Forschungs- und Entwicklungstrends:

- **Edge Computing und KI im Fahrzeug:** Immer mehr Analyse- und Verarbeitungsfunktionen werden direkt ins Fahrzeug verlagert (Edge AI), um Latenzzeiten zu reduzieren und datenschutzrelevante Informationen lokal zu verarbeiten.
- **Continuous Integration/Continuous Deployment (CI/CD) für Fahrzeuge:** Adaption von DevOps-Prinzipien aus der Softwareentwicklung für den Fahrzeugbereich, um schnellere Release-Zyklen und eine kontinuierliche Bereitstellung von Software-Updates zu ermöglichen.
- **Standardisierung und Ökosysteme:** Zusammenarbeit in der Industrie zur Schaffung offener Standards und Plattformen, die eine breitere Interoperabilität und ein vielfältigeres Software-Ökosystem fördern. Beispiele sind der Ansatz von Eclipse SDV oder Projekte wie AUTOSAR Adaptive.
- **Sicherheit durch Design (Security by Design):** Integration von Sicherheitsaspekten von Anfang an in den Entwicklungsprozess von Software und Update-Mechanismen.
- **Dynamische Workload-Verteilung:** Optimierung der Ressourcennutzung durch dynamische Zuweisung von Workloads zu den verfügbaren Rechenressourcen im Fahrzeug.

Zusammenfassend lässt sich sagen, dass Update und Monitoring im SDV-Kontext weit über die traditionelle Fehlerdiagnose hinausgehen. Sie sind integrale Bestandteile eines intelligenten, vernetzten Fahrzeugs, das sich kontinuierlich weiterentwickeln und an neue Anforderungen anpassen kann, um sowohl die Benutzererfahrung als auch die Sicherheit und Effizienz zu verbessern.

3.3.1.2 AP 3.2.1: Monitoring

3.3.1.2.1 Definitionen Monitoring & CVC Use Cases Monitoring

Im Rahmen der Aktivität WP3.2.1 Monitoring wurde eine Unterteilung der Begrifflichkeiten Monitoring, Diagnostik und Logging vorgenommen. Dies soll weitere Diskussionen und Ausführungen erleichtern.

- **Monitoring:**
Proaktive, automatisierte Beobachtung vorhandener Systemparameter
- **Diagnostik:**
Ausgewählte, interaktive Analyse aufgetretener Fehler oder Symptome
- **Logging:**
Aufzeichnung und Nachverfolgung von Systemfehlern und -variablen

In Zusammenarbeit mit den Projektpartnern wurde eine Übersicht (siehe Abbildung 23) zum Thema Monitoring und Logging erstellt, mit dem Ziel verschiedene Handlungsstränge innerhalb des AGEDA Projekts zu identifizieren und zu bündeln. Die Analyse zeigte, dass Überschneidungen zu vielen Arbeitspaketen bestehen und ein Monitoring sowie Logging an verschiedenen Stellen innerhalb des AGEDA Frameworks möglich, angedacht bzw. notwendig ist.

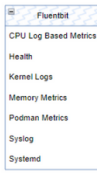
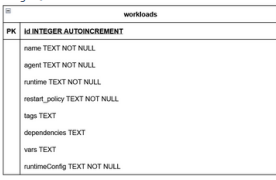
No.	Description	Responsible Team or WP	Purpose	Details	Remark
2	Platform Services: <ul style="list-style-type: none"> Logging 	WP 2.1.4 AGEDA Framework	Troubleshooting, Monitoring and Audit and Compliance	<ul style="list-style-type: none"> Possible solution with Fluentbit 	
	Node Monitoring <ul style="list-style-type: none"> Dynamic Reconfiguration Hackathon 	WP2.3 Dynamic Rekonfiguration	Detection problems on a node and shift workload to another node	The decision if an action needs to be taken has to be done by another component, e.g., the Composer detects a too high temperature and relocates a workload to another node.	Where does it take place?
3	Edge Enabler Services: <ul style="list-style-type: none"> Logging & Monitoring Workload Database 	WP 2.1.4 AGEDA Framework	Monitoring for Edge Operation	<ul style="list-style-type: none"> Persistent storage of workload requests and (possibly) states 	
	Vehicle API Server / Access Token Operator / Access Token Server		<ul style="list-style-type: none"> Storage of Vehicle Access 	<ul style="list-style-type: none"> ANKAIS orchestrator control interface to get the current state (e.g., start, stop,...) of the services The Workload Database provides a persistency layer for running workloads. Using SQLite 	unclear whether Platform or Edge

Abbildung 23: Auszug der Übersicht zum Thema Monitoring und Logging von der Confluence-Seite ([Link](#))

In enger Kooperation mit WP1.1 wurden Use Cases speziell zum Themenkomplex Monitoring definiert, welche dann wiederum bei der Implementierung der Demonstratoren berücksichtigt wurden. Das Zusammenspiel mit WP 1.1 und WP 1.3 ist in Kapitel 3.1 beschrieben.

3.3.1.2.2 Bewertung der Standards UDS, SOVD

Nach einer Bewertung von Standards für die Entwicklung des AGEDA-Frameworks und von Applikationen aus der Perspektive der Überwachung und Diagnostizierbarkeit, wurde der SOVD-Standard (Service-Oriented-Control) von der ASAM-Organisation (<https://www.asam.net/standards/detail/sovd/>) als am besten geeigneten für das AGEDA-Framework von AVL DiTest ausgewählt.

Dies liegt an seiner Fähigkeit, die neuen Architekturen von Software Defined Vehicles (SDVs) mit verbundenem Cloud-Ökosystem wie Gaia-X zu integrieren und dynamische Änderungen in der Fahrzeugsoftware effizient zu bewältigen. Der SOVD-Standard bietet eine flexible und skalierbare Lösung, die den Anforderungen moderner Fahrzeugdiagnosesysteme gerecht wird.

Heutige Diagnosesysteme sind ECU-zentriert und basieren stark auf dem Unified Diagnostic Services (UDS) Protokoll. UDS ist ein statischer Ansatz für Diagnosen, der im Gegensatz zu den dynamischen Softwareaufgaben steht. Daher wäre die Erweiterung des UDS-Protokolls für die Diagnoseanforderungen von HPCs nicht flexibel genug, um die notwendigen Software-Analyseanforderungen zu erfüllen. UDS-basierte ECUs können über den Classic Diagnostic Adapter (CDA) integriert werden.

ASAM SOVD basiert auf HTTP/REST, JSON und OAuth. Der Schwerpunkt von ASAM SOVD liegt auf der Entwicklung der API, während für die Implementierung Diskussionen mit AUTOSAR begonnen haben. Der schematische Aufbau ist in Abbildung 24 dargestellt.

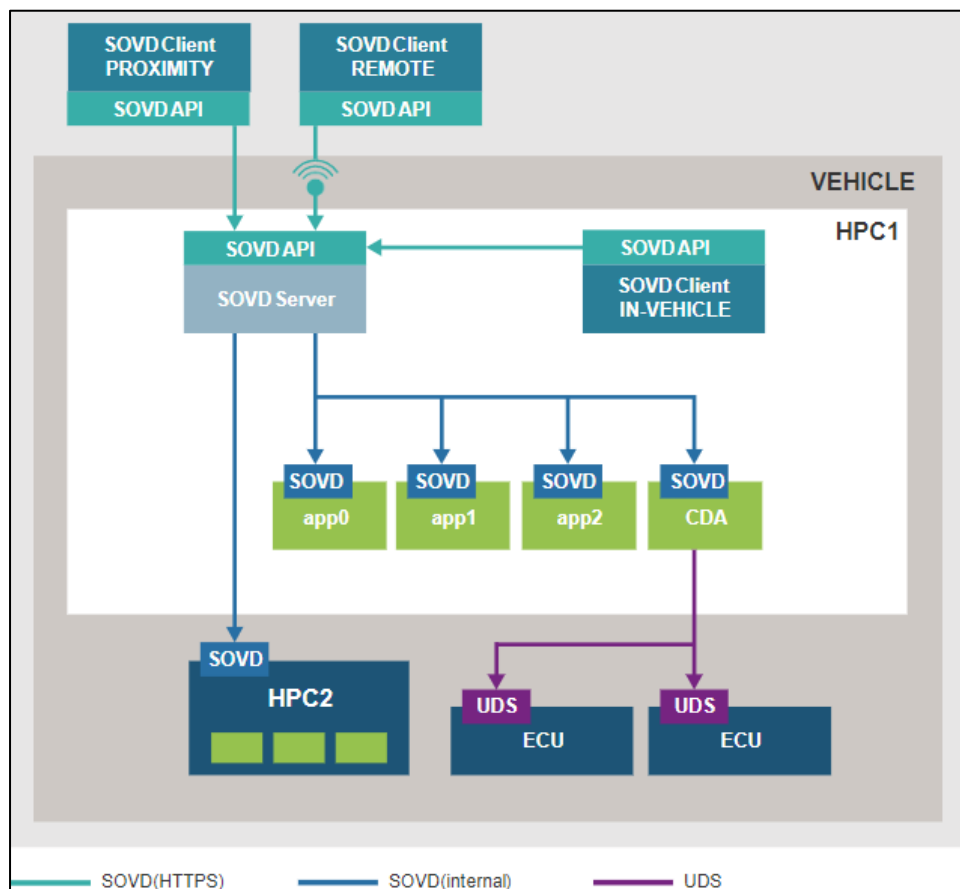


Abbildung 24: ASAM SOVD Schema (Quelle: <https://www.asam.net/standards/detail/sovd/>)

Aus dem Standard lassen sich folgende Anwendungsbereiche ableiten:

- Diagnostische Kommunikation zu HPCs und ECUs (remote, proximity, onboard)
- Software-Updates
- Logging
- Zugang zu Systeminformationen
- Dynamisches Erkennen von Inhalten

Angelehnt an den Adaptive Autosar Standard R23-11 (Explanation of Service-Oriented Vehicle Diagnostics) dient ein zentrales SOVD-Gateway, den sogenannten SOVD-Server, als zentraler Knotenpunkt und leitet Anfragen an die entsprechenden internen SOVD-Endpunkte weiter. Die Weiterleitung erfolgt basierend auf dem Entitätsteil in der URI der SOVD-Anfrage.

Die Kommunikation zu dem SOVD-Client über die REST-API kann remote, proximity (im selben Netzwerk bzw. direkt am Fahrzeug) oder onboard (im Fahrzeug) erfolgen. Der Remote Client ist die einzige Lösung, die die Anforderungen eines Software-definierten Fahrzeugs mit Edge-to-Cloud-Architektur wie dem AGEDA-Framework erfüllt: globale Skalierbarkeit, kontinuierliche Softwarepflege, Cloud-Integration, Kosteneffizienz und Sicherheit. Daher steht dieser auch im Fokus der Umsetzung für dieses Projekt.

Wie das SOVD-Konzept für AGEDA konkret implementiert werden kann, wurde [2] veröffentlicht und ist in Kapitel 3.3.1.2.4 beschrieben.

3.3.1.2.3 Möglichkeiten zur Integration des SOVD-Servers in das AGEDA-Framework

Der SOVD-Server ist die zentrale Komponente im Fahrzeug. Im Standard ist die REST-API definiert, über die die Clients mit dem Server bidirektional kommunizieren. Wie der SOVD-Server fahrzeugseitig integriert wird und die Schnittstellen zu den Komponenten und Applikationen, im Folgenden als AGEDA Workloads bezeichnet, implementiert werden, sind im ASAM-Standard nicht definiert.

Im Projekt werden drei grundsätzliche Varianten betrachtet und mittels der Methode Architecture Decision Record (ADR) bewertet. ADR ist ein leichtgewichtiges Vorgehen, um Architekturentscheidungen in Softwareprojekten nachvollziehbar zu dokumentieren. Die Varianten werden im Folgenden kurz beschrieben:

1. SOVD-Server als AGEDA Workload über VSS
2. SOVD-Server als AGEDA Plugin bzw. AGEDA Building Block
3. SOVD-Server außerhalb des AGEDA Frameworks

SOVD-Server als AGEDA Workload

Die Erste Variante beschreibt die Implementierung des SOVD-Servers in einem SOVD-Server Workload. Die systematische Darstellung ist in Abbildung 25 zu sehen.

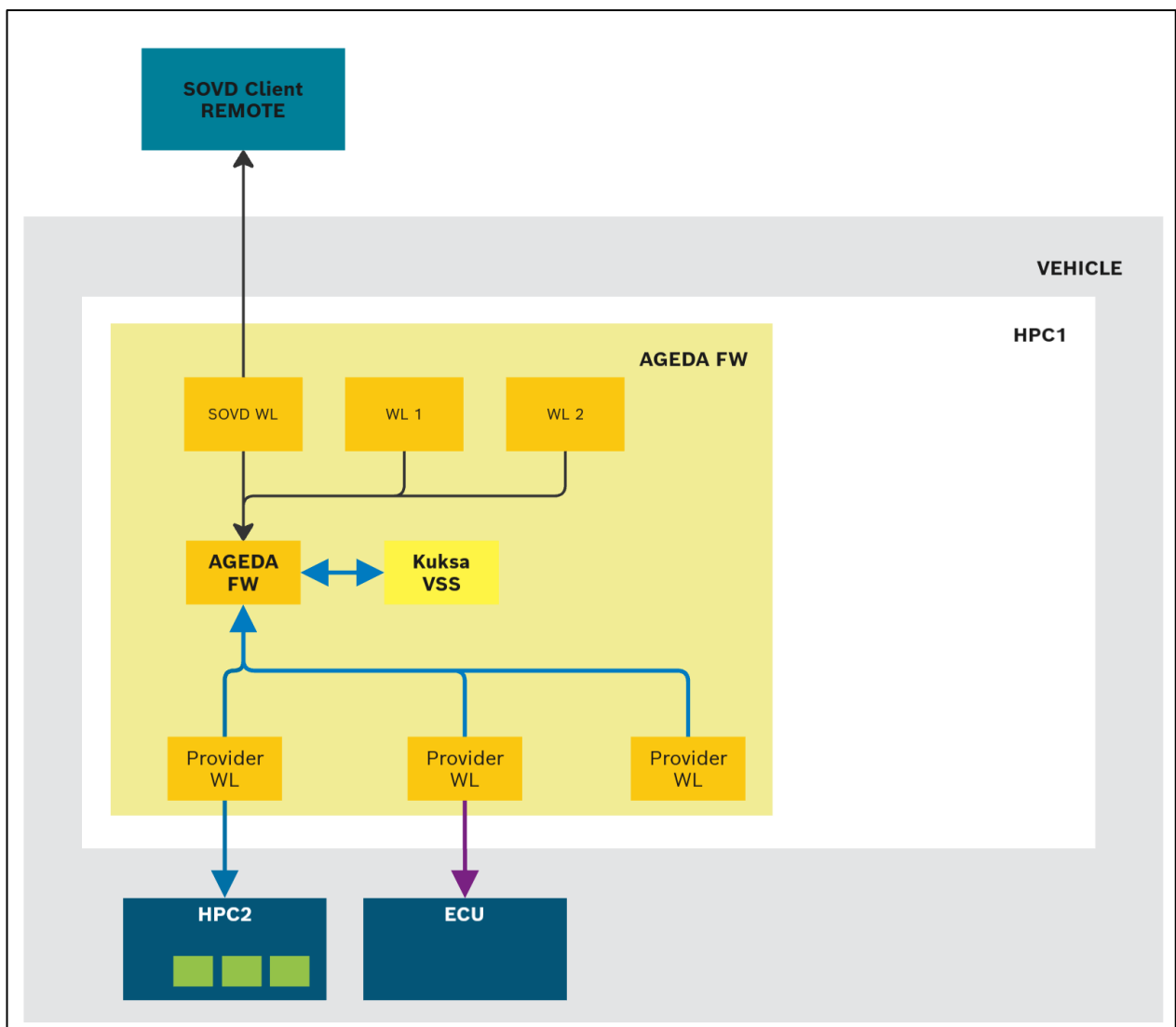


Abbildung 25: SOVD-Server als AGEDA Workload

Bei dieser Variante wird der SOVD-Server als ein Workload ins AGEDA Framework integriert. Workloads ist es nicht erlaubt untereinander zu kommunizieren bzw. Daten auszutauschen. Ein Workload darf aber theoretisch auf alle Signale über den KUKSA Databroker zugreifen, was über die Verifiable-Credentials geregelt wird. Diese Credentials gewährleisten eine sichere und überprüfbare Autorisierung der Zugriffe der Workloads auf fahrzeugbezogene Informationen.

Es ist auch möglich bidirektional mit KUKSA Databroker zu kommunizieren, somit könnten Workloads oder Komponenten ihre Diagnoseinformationen bereitstellen. Der KUKSA Databroker stellt eine Abstraktionsschicht bereit, um auf Fahrzeugsignale zuzugreifen, ohne die Details der Fahrzeugarchitektur kennen zu müssen. Er nutzt dabei das COVESA VSS (Vehicle Signal Specification) als Datenmodell.

Der SOVD-Server-Workload liefert auf der einen Seite eine Schnittstelle zur Übersetzung an VSS, um innerhalb des Fahrzeugs zu kommunizieren und auf der anderen seine REST-API zur Kommunikation mit dem Remote Client. Eine vollständig cloud-basierte Installation des SOVD-Server-Workloads wäre somit ebenfalls möglich, was den Cloud-Native-Ansatz verfolgt. Die Vor- und Nachteile sind in Tabelle 3 dargestellt.

Tabelle 3: Vor- und Nachteile SOVD als AGEDA Workload

Vorteile	Nachteile
Autorisierung mittels Verifiable-Credentials AGEDA Framework Ansatz wird verfolgt	Kuksa/VVS - SOVD Interface muss entwickelt werden
Cloud Native Ansatz wird verfolgt	Einschränkungen durch VSS bzgl. Diagnose Use Cases

SOVD-Server als AGEDA Building Block

Variante zwei platziert den SOVD-Server als Building Block im AGEDA Framework, vergleichbar zur Implementierung des KUKSA Databrokers. Dies würde in die bereits bestehende Architektur des AGEDA Frameworks passen. Building Blocks sind native, statische Komponenten, die vom OEM oder Zulieferer bereitgestellt und über Updates aktualisiert werden können. Sie sind optional – das Framework läuft auch ohne diese Komponenten. Charakteristische Merkmale für Building Blocks sind:

- Containerisierte Applikationen, die nativ installiert sind
- Integration in das AGEDA-Framework über den Framework Orchestrator
- Dadurch Zugriff auf User-Workloads und Framework Komponenten
- Empfohlenes Interface: gRPC für effiziente Kommunikation, aber auch sonst keine Einschränkungen

Eine solche Implementierung des SOVD-Servers ist in Abbildung 26 zu sehen.

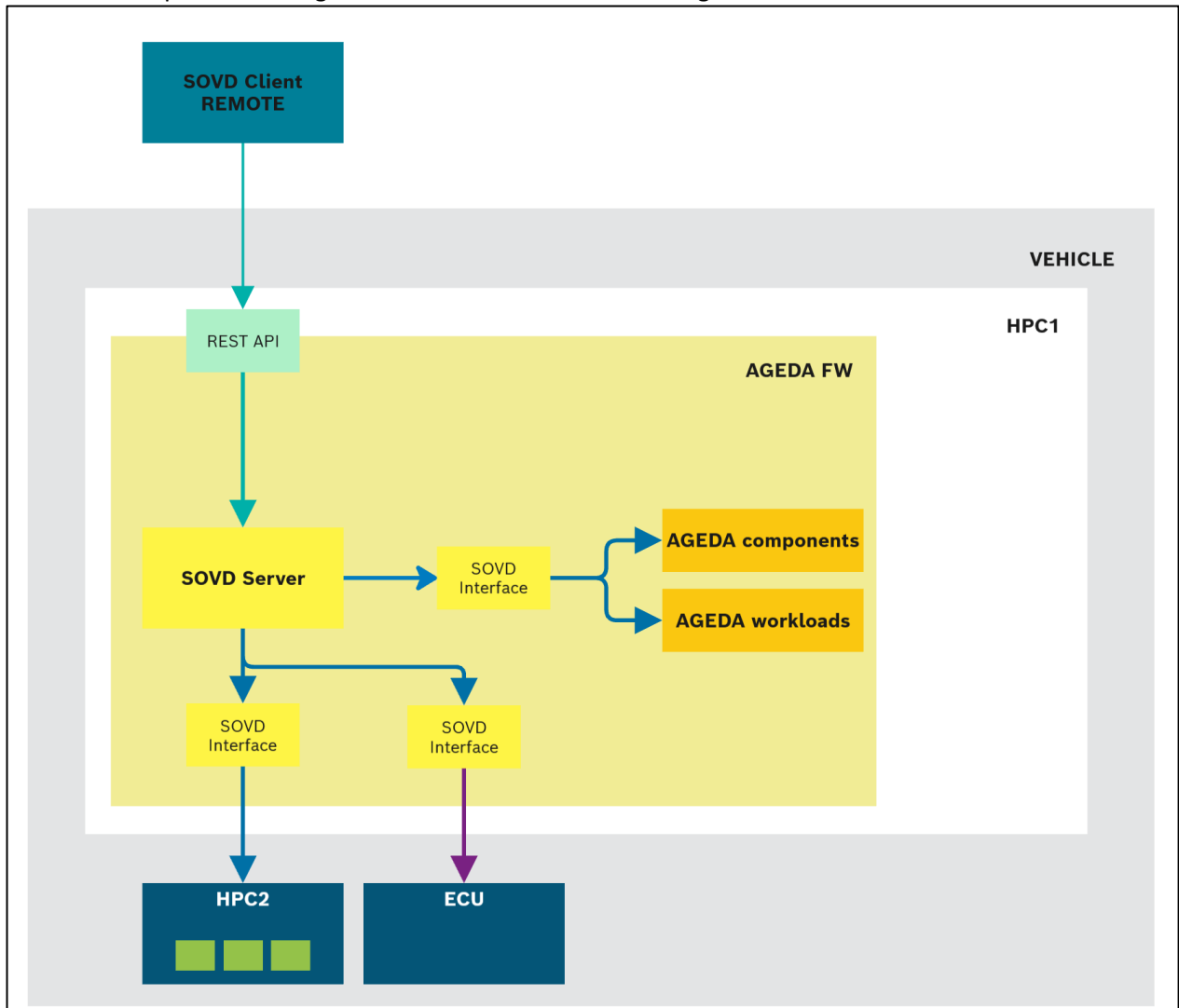


Abbildung 26: SOVD-Server als AGEDA Building Block

Der SOVD-Server ist Building Block im AGEDA-Framework und kann seine REST-API zum Remote Client aufbauen. Um eine vollständige Integration in die SOVD-Architektur zu gewährleisten, muss für jeden Workload und jede Fahrzeugkomponente eine standardisierte SOVD-API implementiert werden. Diese API dient als Schnittstelle, über die die jeweiligen Workloads und Komponenten ihre relevanten Informationen und Funktionen dem zentralen SOVD-Server bereitstellen. Erst durch diese Anbindung kann die gesamte SOVD-Funktionalität genutzt werden. Die Workloads und Komponenten, die über eine solche API verfügen, gelten als „SOVD-ready“. Die Integration von SOVD hat keinen Einfluss auf die bestehenden Berechtigungsprozesse der Workloads über die VCs. Die Vor- und Nachteile sind in Tabelle 4 dargestellt.

Tabelle 4: Vor- Nachteile SOVD als AGEDA Building Block

Vorteile	Nachteile
Autorisierung mittels Verifiable-Credentials: AGEDA Framework Ansatz wird verfolgt	Aufwand bei der Implementierung der SOVD-API für die Workload und Komponenten
Cloud Native Ansatz wird verfolgt	Aufwand der Containerisierung des SOVD-Servers

SOVD-Server außerhalb des AGEDA Framework

Die letzte Variante platziert den SOVD-Server außerhalb des AGEDA Frameworks. Die Anbindung an das AGEDA Framework geschieht mittels dedizierten SOVD-Interface, Abbildung 27.

In diesem Szenario befindet sich der SOVD-Server außerhalb des AGEDA-Frameworks. Dies entspricht dem klassischen Ansatz, bei dem beide Welten – SOVD und AGEDA-Framework – voneinander getrennt und unabhängig bleiben. Durch diese Entkopplung erhält der SOVD-Server größere Freiheiten in Bezug auf Architektur, Implementierung und Weiterentwicklung, ohne durch die Vorgaben des AGEDA-Frameworks eingeschränkt zu werden.

Ein SOVD-Interface, welches noch im Detail zu definieren ist, verbindet AGEDA-Workloads oder -Komponenten mit dem SOVD-Server, sodass sie ihre Diagnosedaten bereitstellen. Die Komponenten außerhalb des Frameworks lassen sich mit den vorhandenen, AGEDA unabhängigen SOVD-Interfaces anbinden. Die Vor- und Nachteile sind in Tabelle 5 dargestellt.

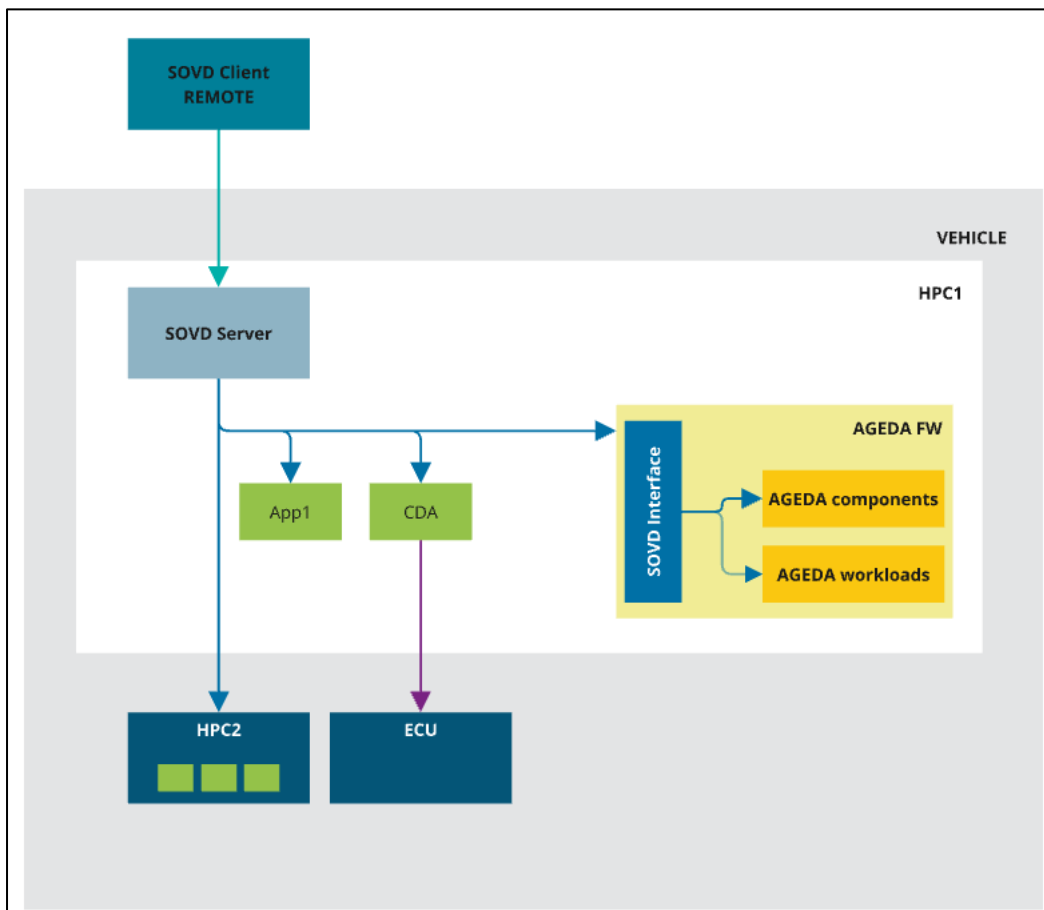


Abbildung 27: Verbindung zum SOVD-Server mittels dedizierten Interface

Tabelle 5: Vor- und Nachteile SOVD außerhalb des AF

Vorteile	Nachteile
Vorhandene Interfaces zu Nicht-AGEDA-Fahrzeugkomponenten	Autorisierung muss SOVD-seitig implementiert werden
Geringere Abhängigkeiten vom AGEDA Framework	SOVD-Interface muss definiert und implementiert werden
	Kein Cloud-native Ansatz: alle SOVD-Komponenten müssen vorinstalliert sein

Fazit und Variantenauswahl

Aufgrund der Vorteile, die das AGEDA-Framework bietet – darunter:

- Sichere Authentifizierung und Verwaltung von Zugriffsrechten
- Cloud-native Architektur für Skalierbarkeit und Flexibilität
- Zentrales Management und Orchestrierung von Diensten und Prozessen

erscheint es sinnvoll, den SOVD in das AGEDA-Framework zu integrieren. Durch diese Kombination lassen sich die Stärken beider Ansätze vereinen: Die standardisierte Diagnose- und Datenzugriffslogik von SOVD wird mit den Sicherheits-, Orchestrierungs- und Cloud-Fähigkeiten des AGEDA-Frameworks ergänzt. Dies schafft eine ganzheitliche, zukunftsfähige Plattform für softwaredefinierte Fahrzeuge.

Innerhalb des AGEDA-Frameworks soll der SOVD-Server als Building Block realisiert werden. Diese Variante bietet entscheidende Vorteile: direkte Zugriffsberechtigungen für Workloads, eine nahtlose Integration in weitere AGEDA-Komponenten sowie die Möglichkeit zur umfassenden Überwachung und Steuerung.

Die Alternative, den SOVD-Server als AGEDA-Workload unter Verwendung von VSS (Vehicle Signal Specification) als Kommunikationsschnittstelle umzusetzen, bringt erhebliche Nachteile mit sich:

- Publish-Subscribe-Paradigma: VSS basiert auf einem Pub-Sub-Modell, bei dem Signale an mehrere Abonnenten verteilt werden, ohne einen direkten Anfrage-Antwort-Mechanismus. Für die Fahrzeugdiagnostik ist jedoch ein Request-Response-Paradigma unerlässlich, da präzise Abfragen und exakte Antworten für eine zuverlässige Fehlererkennung und -behebung erforderlich sind.
- Fehlende Quellinformationen: VSS-Signale enthalten keine eindeutige Herkunftsangabe, was die präzise Identifikation von Problemen während der Diagnostik erheblich erschwert.
- Komplexität durch Versionsvarianten: Diagnosen erfordern die Berücksichtigung unterschiedlicher Fahrzeugmodelle und Softwareversionen. Das Pub-Sub-Modell von VSS adressiert diese Anforderungen nicht ausreichend.
- Ungeeignet für Logging: Aufgrund fehlender Quellinformationen und des Pub-Sub-Modells ist VSS für detailliertes Logging ungeeignet, wenn eine lückenlose Nachverfolgung der Signalherkunft notwendig ist.
- Unzureichend für Software-Updates: Für Updates sind präzise Nachverfolgung und Versionskontrolle essenziell. Das VSS-Modell unterstützt diese Anforderungen nicht effektiv.

VSS ist zwar effizient für die allgemeine Signalverteilung, erfüllt jedoch nicht die Anforderungen einer präzisen und reaktionsschnellen Diagnostikumgebung. Dennoch existieren spezifische Use Cases, in denen eine Kombination von VSS und SOVD sinnvoll sein kann (vgl. Böhlen, Diagnosetagung 2025 [1]).

Für die Umsetzung als Building Block muss der SOVD-Server containerisiert werden, was technisch problemlos möglich ist. Die zentrale Herausforderung liegt in der Definition einer geeigneten Schnittstelle zu den Workloads und Komponenten. Diese wird in Kapitel 3.3.1.2.4 detailliert beschrieben.

Die Variantenauswahl wurde in einem ADR im gemeinsamen Confluencespace [3] des Projekts dokumentiert, siehe Abbildung 28.

Context and problem statement	Considered Options	Decision Outcome	Status	Date (opened)	Who (created)
Location of SOVD server	within AGEDA FW outside of FW	Inside of FW , because of benefits regarding: <ul style="list-style-type: none"> • authentication & access rights • cloud native architecture and • overall managing of AGEDA-FW 	CLOSED	21.10.2024	@ Arthur Ehler
Variant of Integration	as Workload with VSS as Interface as AGEDA Plug-In	as AGEDA Plug-In , because <ul style="list-style-type: none"> • VSS is effective for general signal distribution, it falls short in the precise and responsive environment needed for vehicle diagnostics • workloads have also limited access rights to AGEDA-FW (logger, Event sourcing etc.) • Plug-In: fits to AGEDA Philosophy, nately installed, full Access rights to AGEDA-FW (via AGEDA role manager) 	CLOSED	27.11.2024	@ David Buch

Abbildung 28: ADR zur SOVD-Integration

3.3.1.2.4 Implementierung Monitoring

Im Folgenden werden die Ergebnisse beschrieben, welche aus der Implementierung des Monitorings ergeben haben. Dies beinhaltet hauptsächlich die Integration des SOVD-Servers aber auch unterstützende Elemente, wie Workloads und Data-Provider. Alle beschriebenen Ergebnisse wurden mit dem Ziel zur Demonstration entwickelt und wurden erfolgreich beim Abschlussevent präsentiert. Abbildung 29 zeigt die Realisierung des AGEDA Framework bei der Demo, wie sie im Bosch Fahrzeug umgesetzt wurde.

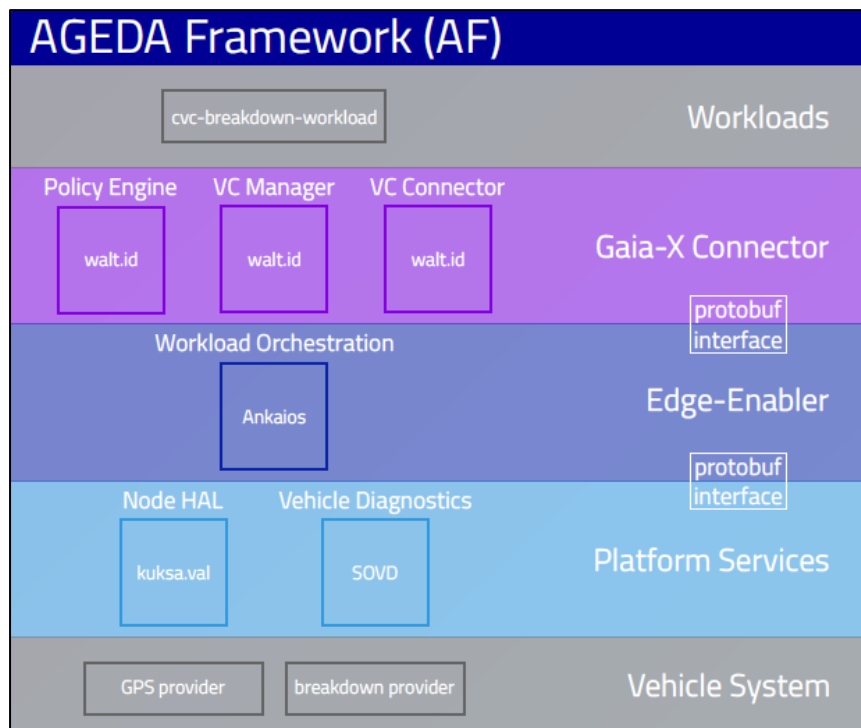


Abbildung 29: Finale AF Architektur Monitoring

Abbildung 29 zeigt die Realisierung des AGEDA Framework bei der Demo, wie sie im Bosch Fahrzeug umgesetzt wurde. Auf unterster Ebene stehen die Platform Services. Hier ermöglicht Kuksa-Komponente die Verbindung zu den Fahrzeugsignalen. Die im Zuge dieses Arbeitspaketes entwickelte SOVD-Komponente befindet sich auf derselben Ebene. Die detaillierte Beschreibung folgt im nächsten Abschnitt.

Die Orchestrierung der Services oder Workloads übernimmt der Ankaio-Workload-Orchestrator auf der Edge-Enabler Ebene. Für die Verifizierung und das Credential Management und um die Services zu starten sind die Komponenten auf der Gaia-X Connector Ebene zuständig. Dazu gehören die Policy Engine, der VC-Manager und der VC Connector.

Konkrete Funktionen der Services und Anbindungen ans Fahrzeug sind im CVC-Breakdown-Workload, sowie im GPS-Provider und dem Breakdown-Provider implementiert.

Die vorliegenden Ergebnisse sind das Produkt einer sehr engen Zusammenarbeit zwischen Bosch und AVL DiTest. Allgemeine Architektur-Entscheidungen wurden gemeinschaftlich erarbeitet und konsentiert. Während der Implementierungsphase trug AVL DiTest die Hauptverantwortung für die SOVD-Bibliothek (libsovd) sowie die Backend-Anbindung des Diagnose-Clients. Bosch konzentrierte sich hingegen auf die Integration in das AGEDA Framework, insbesondere im Hinblick auf die Workloads. Zur besseren Verständlichkeit der Arbeitsergebnisse sind nachfolgend die gemeinsamen Ergebnisse dargestellt.

Integration des SOVD-Servers ins AGEDA-Framework (Leitung AVL DiTest)

Wie das SOVD-Konzept für AGEDA implementiert werden kann, wurde in (Röper, Buch, Fiedler, & GmbH), 05/2024) veröffentlicht und ist im Folgenden auszugsweise beschrieben und in Abbildung 30 dargestellt.

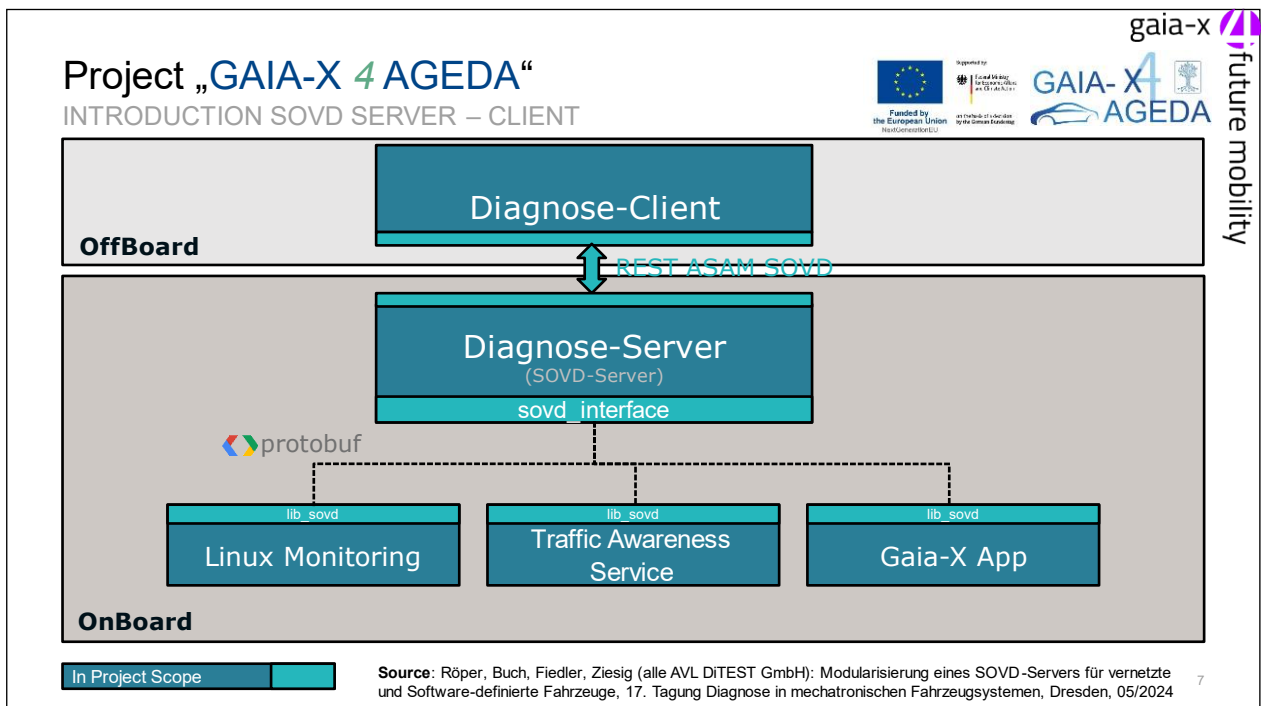


Abbildung 30: SOVD-Architektur für AGEDA

Entgegen der Implementierung im Adaptive Autosar geht das SOVD-Integrationskonzept für AGEDA jedoch einen Schritt weiter und es wird nicht davon ausgegangen das jede Applikation wieder ein SOVD-Server implementiert. Vielmehr wird für die Applikationen, wie das Linux Monitoring oder der Traffic Awareness Service, eine Library (libsovd) zur Verfügung gestellt. Die Library dient als API für Applikationen um Datenpunkte im SOVD-Server verfügbar zu machen.

Innerhalb des AGEDA-Frameworks fungiert die SOVD-Bibliothek (libsovd) als zentrale API für AGEDA-Apps. Diese Bibliothek ermöglicht es den Applikationen, Diagnosedaten über den SOVD-Server bereitzustellen und abzurufen. Durch die Nutzung dieser API wird eine standardisierte und effiziente Kommunikation zwischen den verschiedenen Softwarekomponenten und den Diagnosesystemen des Fahrzeugs sichergestellt.

Die Nachrichten werden über Protocol Buffer serialisiert. Aus der Erfahrung der AVL DiTEST im Bereich von OnBoard-OTA-Lösungen ist dieser Ansatz wesentlich praxistauglicher da Protocol Buffers Implementierung wesentlich schlanker sind, weniger Abhängigkeiten haben und geringere Anforderungen an das System Stellen.

Neben dem SOVD-Server-Gateway entsteht ein Linux Monitoring als Beispielimplementierung einer Applikation, welche die Auslastung der AGEDA-ECU in Diagnosedaten verfügbar macht. Wird ein neuer Service oder Applikation wie der Traffic Awareness-Service aus dem Gaia-X-Datenraum heruntergeladen, können neue Datenpunkte und Fehlercodes von hochdynamischen SOVD-Clients jederzeit im neuen Schema der Endpunkte erkannt werden und stehen somit sofort der Diagnoseapplikation zur Verfügung.

Der SOVD-Server stellt nach dem Standard eine REST-API (Representational State Transfer) dar. Die so gesammelten Informationen werden vom SOVD-Client in der Cloud bzw. Offboard (Abbildung 30) über eine http-basierte REST-Schnittstelle abgerufen. Wie genau die Verbindung zwischen Server und Client aufgebaut wird, ist aber weiter nicht spezifiziert. Die Fahrzeuge sind typischerweise via LTE-Sim mit dem Internet verbunden. Dabei wird dem SOVD-Server nur eine geteilte IP-Adresse zugeordnet mit entsprechenden Restriktionen für eingehende Kommunikation. Die Herausforderung besteht darin, den SOVD-Server fahrzeugseitig zu erreichen. In einem ersten Schritt und für die Implementierung in der Demo 2024 (siehe Kapitel 3.1.2.3) wird die Reverse SSH (Secure Shell) Tunneling Methode gewählt, da sie sicher und praktikabel ist und dem SOVD-Standard entspricht. Hierfür baut ein weiterer Server mit HTTP-Firewall den SSH-Tunnel zum SOVD-Server im Fahrzeug auf und öffnet den Port für den SOVD Remote-Client. Die finale Implementierung wird aber mit MQTT durchgeführt, aufgrund der Skalierbarkeit und Effizienz (Flottenanwendung) und gleichzeitiger guter Firewall-Freundlichkeit.

Viele IoT-Geräte befinden sich hinter NAT-Firewalls und verfügen nicht über öffentliche IP-Adressen, wodurch eingehende HTTP-Verbindungen aus dem Internet nicht möglich sind. Die direkte Nutzung von HTTP für Remote-Diagnose ist daher nicht praktikabel. Dies erfordert eine Lösung, die sowohl die Einschränkungen von NAT-Umgebungen überwindet als auch eine sichere, skalierbare Kommunikation ermöglicht. Der Secure MQTT Connector adressiert diese Probleme durch die Nutzung des MQTT-Protokolls, einem leichtgewichtigen Publish/Subscribe-Mechanismus, der speziell für IoT-Anwendungen entwickelt wurde. Die Kernfunktionen sind die Protokoll-Transformation von HTTP-Aufrufen in MQTT-Nachrichten, das zentrale Zertifikatsmanagement, die Sammlung von Diagnosedaten und Log-Meldungen sowie die Unterstützung für den Dateiaustausch zwischen Fahrzeug und Cloud. Das Protokoll ist cloud-agnostisch und kann universell implementiert werden. Die Lösung wurde erfolgreich in mehreren Projekten eingesetzt und bietet eine deutlich einfachere Handhabung im Vergleich zu früheren SSH-Tunnel-Ansätzen. Herausforderungen bestehen aktuell bei hoher Last sowie bei der Übertragung großer Datenmengen, wofür zusätzliche Protokollerweiterungen geplant sind. Im Vergleich zu SSH-Tunneln bietet der Secure MQTT Connector zahlreiche Vorteile: Erstens ist die Skalierbarkeit deutlich höher, da MQTT für viele parallele Verbindungen optimiert ist. Zweitens ist das Protokoll effizienter und ressourcenschonender, was für IoT-Geräte entscheidend ist. Drittens umgeht MQTT NAT-Probleme durch ausgehende Verbindungen, während SSH-Tunnel oft zusätzliche Konfigurationen erfordern. Viertens bietet MQTT integrierte Sicherheitsmechanismen wie TLS und Zertifikatsmanagement, während SSH manuelles Schlüsselmanagement voraussetzt. Fünftens ermöglicht MQTT eine flexible, asynchrone Kommunikation über Publish/Subscribe, wohingegen SSH-Tunnel rein verbindungsorientiert sind. Schließlich erlaubt MQTT integrierte Monitoring- und Logging-Funktionen, die bei SSH zusätzliche Tools erfordern.

Auf dem Remote-Client lassen sich Diagnose- und Monitoring Informationen anzeigen. Ein OEM, Service-Provider, Flottenmanager oder Service Techniker, kann damit gezielt die Interaktion eines auf dem Fahrzeug ausgeführten Gaia-X-Service mit dem Fahrzeug selbst monitoren und diagnostizieren. Abbildung 31 zeigt die Benutzeroberfläche des über einen Internetbrowser erreichbaren Remote SOVD-Clients.

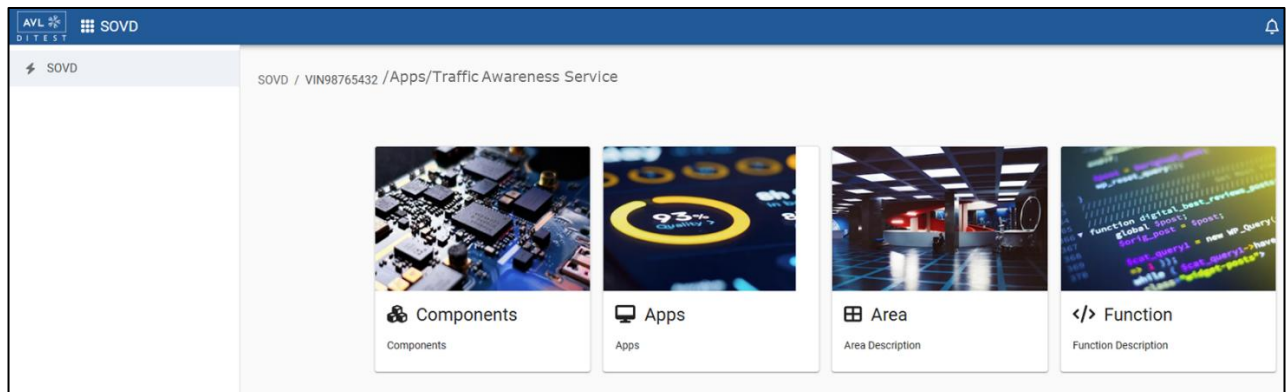


Abbildung 31: Browser-Benutzeroberfläche des AVL DiTEST SOVD Remote Clients

Workloads (Leitung Bosch)

Im Zuge des Arbeitspaketes und der Teilnahme an Fahrdemos wurden mehrere Services oder Workloads entwickelt und ebenfalls im Fahrzeug getestet. Diese dienen dazu die Entwickelte SOVD-Komponente zu demonstrieren und zu testen, waren also ein essenzieller Bestandteil bei der Entwicklung und sollen folglich nachfolgend beschrieben werden.

CVC-Breakdown-Workload (Leitung Bosch)

Der CVC-Breakdown-Workload wurde für die Fahrdemos Level 3 und 4 als Teil des Traffic-Awareness-Service entwickelt und außerdem dazu genutzt das SOVD-Monitoring im Feld zu erproben. Der Quelltext ist hier zu finden: [Gitea cvc-breakdown-workload](#). Die Applikation wurde in Python realisiert. Im Folgenden wird sein Aufbau und Funktionalität beschrieben.

Eine Übersicht des Workloads ist in Abbildung 32 zu sehen. Der Service wird über das AF gestartet und orchestriert. Verbindungen zum Vehicle-Layer sind über den Kuksa Databroker sowie den SOVD-Server realisiert. Mittels MQTT ist er mit einem MQTT-Broker verbunden.

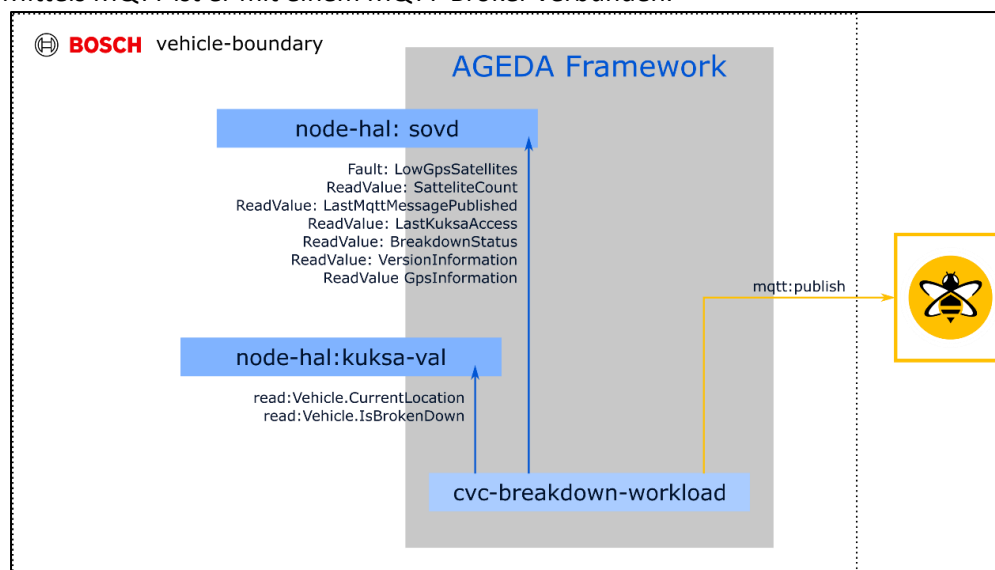


Abbildung 32: Übersicht CVC-Breakdown-Workload

Kontinuierlich werden GPS-Position und Fahrzeugausrichtung (Vehicle.CurrentLocation) sowie der Fahrzeugstatus (Vehicle.IsBrokenDown) vom Kuksa Databroker abgeholt. Diese Informationen werden dann in eine MQTT-Nachricht formatiert und an den Broker versendet. Die anderen Teilnehmer des Traffic-Awareness-Service können diese Information dann für eine optimierte Routenplanung nutzen. Eine Übersicht der Teilnehmer des Service ist in Abbildung 33 zu sehen.

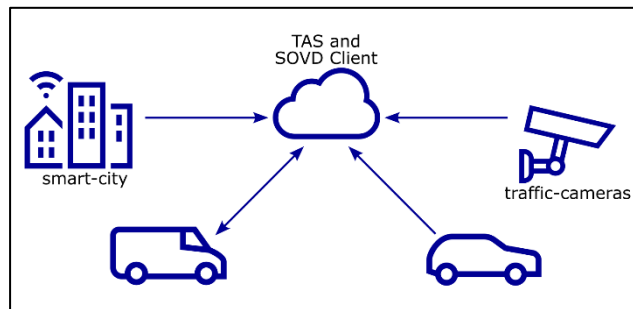


Abbildung 33: Teilnehmerübersicht Traffic-Awareness-Service

Das nachfolgende Bild (Abbildung 34) zeigt eine Ansicht der Birds-Eye-GUI, welche den Service und den Anteil des CVC-Breakdown-Workload zeigt. Das Bosch-Fahrzeug sendet den Status broken_down, worauf das DLR-Fahrzeug eine optimierte Route vorgeschlagen bekommt.

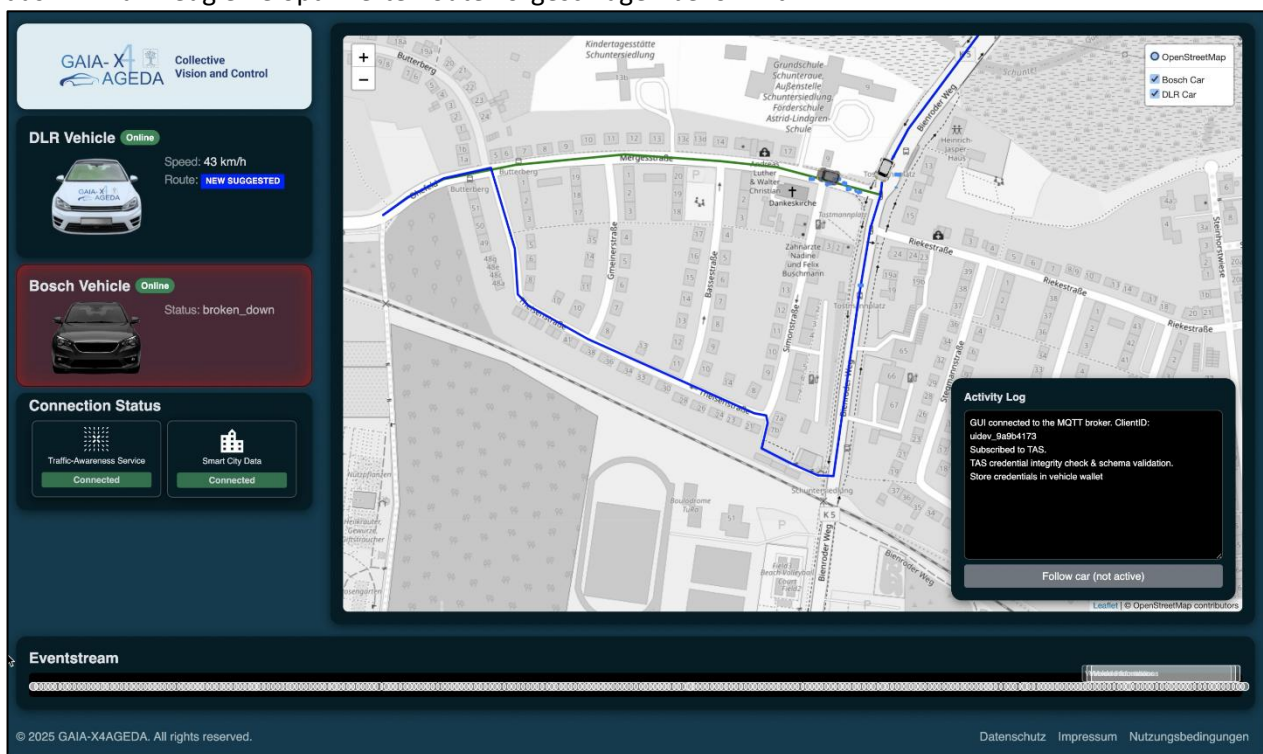


Abbildung 34: Momentaufnahme Birds-Eye-GUI

Die gleichen Informationen, sowie Metainformationen werden dem SOVD-Server bereitgestellt. Zusätzlich meldet der Service dem Server einen Fehler, sollte die GPS-Verbindung aufgrund zu wenig vorhandener Satelliten schwächeln (Fault: LowGpsSatellites). Eine detailliertere Beschreibung der Anbindung eines Workloads mittels SOVD wird im Abschnitt für den nächsten Workload beschrieben.

SOVD-Example-Workload (Leitung Bosch)

Der SOVD-Example-Workload dient als Beispiel für die Anbindung des Monitorings über SOVD an einen Workload. Dieser wurde in Python entwickelt. Der Quellcode ist zu finden unter: [Gitea example-workloads](#). Eine Anwendung hat zwei Möglichkeiten mit dem SOVD-Server zu kommunizieren, über bereitgestellte Read-Values oder über Faults (Fehler). Ist ein Fehler implementiert, so kann die Applikation diesen melden oder bereinigen. Zur Bereitstellung von Read-Values implementiert die App Callback Funktionen, welche auf Nachfrage diese an den Server übermitteln.

Die Implementierung in Python ist mit dem Paket cffi ([pypy: cffi](#)) durchgeführt worden. Für eine erfolgreiche Anbindung braucht jeder Workload individuelle Ressourcen. Diese sind in Tabelle 6 aufgelistet und beschrieben.

Tabelle 6: Notwendige Ressourcen für eine SOVD-Anbindung

Ressource	Beschreibung
<i>sovd_example_workload.sdl</i>	Konfigurationsdokument zur Definition der Read-Values und Fehler. Dieses wird benutzt, um die C-Ressourcen zu generieren.
<i>libsovd.h</i>	C-Header, zu den bereitgestellten Funktionen in der libsovd
<i>libsovd.so</i>	Shared Object File, zu den bereitgestellten Funktionen in der libsovd

Diese Anbindung kann nun für beliebige Workloads realisiert werden. Vorab ist nur zu wissen, Welche Signalwerte und Fehler gewünscht sind und dann müssen die Callbacks innerhalb der Applikation bedient werden. Optimal funktioniert das, wenn die SOVD-Anbindung gleich zu Beginn der Entwicklung berücksichtigt wird. Eine spätere Anbindung ist aber auch möglich, wie im Beispiel des Collision-Warning-Service, wie im nächsten Kapitel beschrieben.

Collision-Warning-Service (Leitung Bosch)

Der Collision-Warning-Service ([Gitea collision-warning-service](#)) wurde ohne Berücksichtigung einer SOVD-Anbindung entwickelt. Somit musste diese im Nachhinein „eingenäht“ werden. Die Hauptapplikation wurde von anderen Projektpartnern entwickelt und durch uns mit der Monitoring Komponente ergänzt.

In Abbildung 35 ist ein Ausschnitt dieser Implementierung zu sehen, genauer: die Definition einer Python-Klasse, die in der Applikationslogik eingebunden werden kann. Die in Abschnitt 0 erläuterten Konzepte sind deutlich zu erkennen. Die Klassendefinition beginnt mit mehreren Attributen, diese entsprechen den vorher erläuterten Read-Values. Außerdem ist zu sehen, wie bei der Initialisierung die in Abbildung 35 aufgelisteten C-Dateien eingebunden werden (Codezeilen 10 bis 20). Die Definitionen der benötigten Callback-Funktionen sind in den Codezeilen 27 bis 55 zu finden.

Die Anbindung des Collision-Warning-Service war ein erfolgreiches Beispiel der Implementierung unserer entwickelten SOVD-Komponente. So fand es auch in der finalen Demo seinen Platz (siehe Abschnitt 3.1.2.4).

```

1  import asyncio
2  import cffi
3
4  class SovdConnector:
5      warn_state: bool = False
6      ego_speed: float = 0.0
7      last_kuksa_update: str = '1991-03-07T12:55:00.000000Z'
8      last_telematic_event_published: str = '1991-03-07T12:55:00.000000Z'
9
10     def __init__(self):
11         self.ffi: cffi.FFI = cffi.FFI()
12         self.run_task: asyncio.Task | None = None
13
14         # Load c-header file
15         with open('/app/src/libsovd.h', 'r', encoding='utf-8') as f:
16             _libsovd_header: str = f.read()
17         self.ffi.cdef(_libsovd_header)
18
19         # Load c-library
20         self.libsovd = self.ffi.dlopen('/app/src/libsovd.so')
21
22     async def run_async(self) -> None:
23         # Initialize connection to sovd
24         self.libsovd.initialize_with_url_port(b'platform-services-pod', 28383)
25
26         # Register read-value callbacks
27         @self.ffi.callback('CollisionWarningServiceWarnState()')
28         def warn_state_callback():
29             _warn_state = self.ffi.new('CollisionWarningServiceWarnState*')
30             _warn_state.state = self.warn_state
31             return _warn_state[0]
32         self.libsovd.register_read_value_callback_collision_warning_service_warn_state(
33             warn_state_callback
34         )
35
36         def sovd_string(value: str):
37             s = self.ffi.new('SovdString*')
38             s.value = self.ffi.new('char[]', value.encode('utf-8'))
39             s.length = len(value.encode('utf-8'))
40             return s[0]
41
42         @self.ffi.callback('CollisionWarningServiceLastKuksaUpdate()')
43         def last_kuksa_update_callback():
44             _last_kuksa_update = self.ffi.new('CollisionWarningServiceLastKuksaUpdate*')
45             _last_kuksa_update.timestamp = sovd_string(self.last_kuksa_update)
46             return _last_kuksa_update[0]
47         self.libsovd.register_read_value_callback_collision_warning_service_last_kuksa_update(
48             last_kuksa_update_callback
49         )
50         try:
51             while True:
52                 self.run_task = asyncio.create_task(asyncio.sleep(60))
53                 await self.run_task
54         except asyncio.CancelledError:
55             pass
56
57     def stop(self) -> None:
58         if self._run_task is not None:
59             if not self.run_task.cancel():
60                 raise asyncio.CancelledError
61

```

Abbildung 35: Code-Snippet SOVD-Connector

Data Provider (Leitung Bosch)

Für die Entwicklung und Realisierung der SOVD-Monitoring Komponente werden zusätzlich sogenannte Data-Provider benötigt. Diese sorgen dafür, dass dem Kuksa Databroker Daten zur Verfügung stehen, welche dann von den Applikationen (siehe Abschnitt 0) benutzt werden. Für die GPS-komponenten wurde der Kuksa-GPS-Provider entwickelt (siehe [Gitea kuksa-gps-provider](#)). Zusätzlich liefert der Kuksa-Breakdown-Provider (siehe [Gitea kuksa-breakdown-provider](#)) Informationen über den Fahrzeugzustand.

3.3.1.3 AP 3.2.2: Update

3.3.1.3.1 Definition Update and Use-Cases

Das Unterarbeitspaket WP3.2.1 verfolgte die Analyse von Update Szenarien im AGEDA Kontext. Im Allgemeinen können sechs verschiedene Update Uses-Cases unterschieden werden:

Tabelle 7: Lister der Update Use-Cases

ID	Use Case	Beschreibung
1	Framework Update	Update des AGEDA Framework
2	Anwendungsupdate	Update einer Anwendung: <ul style="list-style-type: none"> • Spezifische Teile einer Applikation aus bereits gestarteten Workloads • Dies beinhaltet das Update eines oder mehrerer Workloads
3	Deployment eines neuen Containers	Deployment eines neuen Containers
4	Optional: Rollback Szenario	Möglichkeit des Rollbacks infolge eines gescheiterten Updates
5	Cloud Updates (Updates der cloud-seitigen Komponenten)	Lediglich der cloudseitige Teil einer Applikation wird geupdated
6	Fahrzeug Updates (OTA update of ECUs - UN R156)	Updates einzelner ECUs im Fahrzeug

Die Projektpartner haben beschlossen sich auf das Update von Applikationen (User-Workloads) im AGEDA Framework zu fokussieren. Hierfür soll die implementierte Plugin-Architektur genutzt werden.

Folgender Use Case soll dabei abgebildet werden:

1. Ein Workload mit Zugriff auf Fahrzeugsignale wird in der Version 1.0 gestartet.
2. Der Workload hat einen Softwarefehler, welche von einem Entwickler über die Visualisierung und den SOVD-Client erkannt wird
3. Das Image des User-Workloads wird korrigiert und eine Version 2.0 veröffentlicht.
4. Die **sicherheitsunkritische** Applikation in der Version 1.0 wird **automatisch** gestoppt.
5. Die fehlerfreie Version 2.0 wird gestartet.

In enger Kooperation mit WP1.1 wurden Use Cases speziell zum Themenkomplex Update definiert, welche dann wiederum bei der Implementierung der Demonstratoren berücksichtigt wurden.

3.3.1.3.2 Konzept Beschreibung

Im Folgenden soll kurz beschrieben werden, welche Konzepte in welchem Umfang in diesem Arbeitspaket erarbeitet wurden. Die Details zu den Konzepten und deren Umsetzung folgen im Kapitel 3.3.1.3.3.

Im Rahmen der Entwicklung des AGEDA Framework entstand die Komponente "Policy Engine". Diese ist Teil des Gaia-X-Connector-Layers und ermöglicht das Holen, Erstellen und Präsentieren von Verifiable-Credentials. Dies wird genutzt, um Service Workloads zu starten siehe: [Gitea Policy-Engine](#). Die Komponente ist mittels walt-id realisiert: [walt.id](#). In seiner aktuellen Implementierung ermöglicht es die beispielhafte Umsetzung des Updates eines Workloads. Diese Umsetzung wird weiter in Abschnitt 0 beschrieben.

Ein Allgemeines Update Konzept für das AGEDA Framework soll in Abschnitt 0 betrachtet werden. Dieses kann als Leitfaden für eine mögliche Implementierung dienen.

Wie in Kapitel 3.3.1.2.2 beschrieben bietet SOVD die Möglichkeit Updates durchzuführen. Da der SOVD-Sever in das AGEDA-Framework integriert wird, bietet sich an ein Konzept für die Nutzung von SOVD für OTA-Remote-Updates von AGEDA-Applikationen abzuleiten.

3.3.1.3.3 Implementierung Updates

Update: Implementation using Policy Engine

Die Policy Engine Teil des Gaia-X-Connector-Layers und ermöglicht das Holen, Erstellen und Präsentieren von Verifiable-Credentials. Dies wird genutzt, um Service Workloads zu starten. In seiner aktuellen Implementierung ermöglichte es eine beispielhafte Update Umsetzung, auch bei der Live-Demonstration. In Abbildung 36 sind die Schritte, die für das beispielhafte Update durchlaufen werden, aufgelistet. Voraussetzung für diese Implementation ist ein sogenanntes Image-Artifactory, welches den ausführbaren Code über die Cloud bereitstellt, sowie einen Katalog, in dem die Verifiable-Credentials gelistet sind.

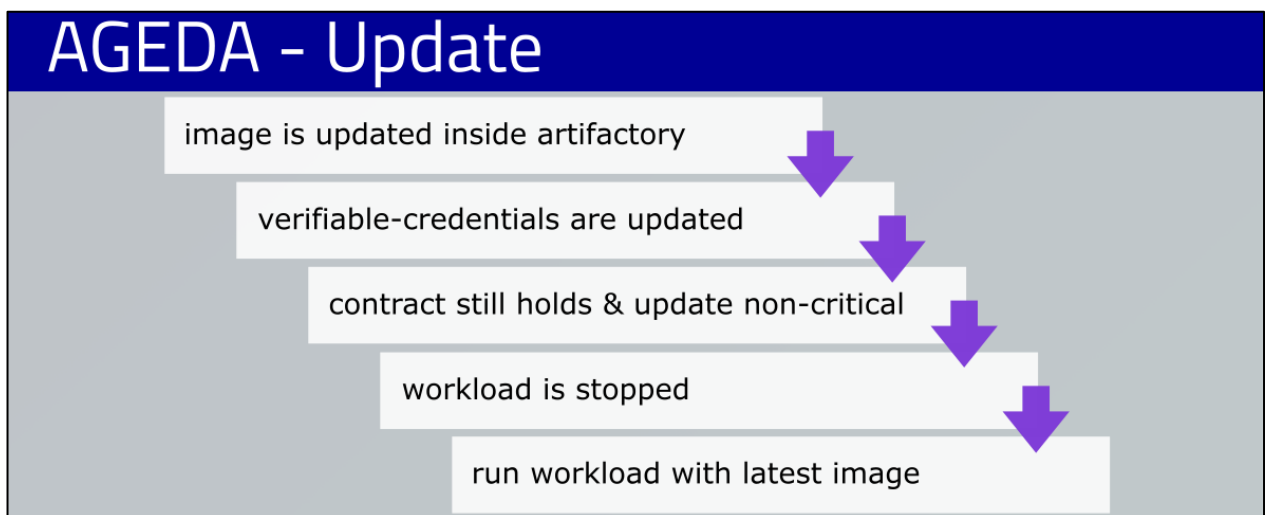


Abbildung 36: Update-Schritte bei Live-Demonstration

Zu Beginn läuft die Version 1.0 eines Workloads, dieser wurde aus dem Image-Artifactory geholt und die Verifiable-Credentials sind gültig. Dann wird ein Bug entdeckt und die Entwickler veröffentlichen eine neue Version, zum Beispiel 1.1. Noch wird das Update nicht ausgeführt. Erst wenn die VC geupdated werden, wird durch die Policy Engine das Austauschen des Workloads gestartet, sofern der Vertrag noch gültig ist und das Update unkritisch eingeschätzt wird. Der veraltete Workload wird nun gestoppt und die neueste Version wird gestartet.

Diese Update Prozedur wurde während der Abschlussdemonstration erfolgreich durchgeführt. Wie beschrieben, wäre diese aber nur für sicherheitsunkritische Software anwendbar. Ein umfangreicheres Konzept muss mehrere Aspekte berücksichtigen. Dies soll im nächsten Abschnitt betrachtet werden.

Update Konzept Allgemein

Im Folgenden soll ein allgemeines Update Konzept für das AGEDA Framework und seine Komponenten in ihrer aktuellen Form beschrieben werden.

Die Strategie beinhaltet Framework Updates, Workload Updates und Updates der Verifiable-Credentials. Abbildung 37 zeigt das Konzept bildlich.

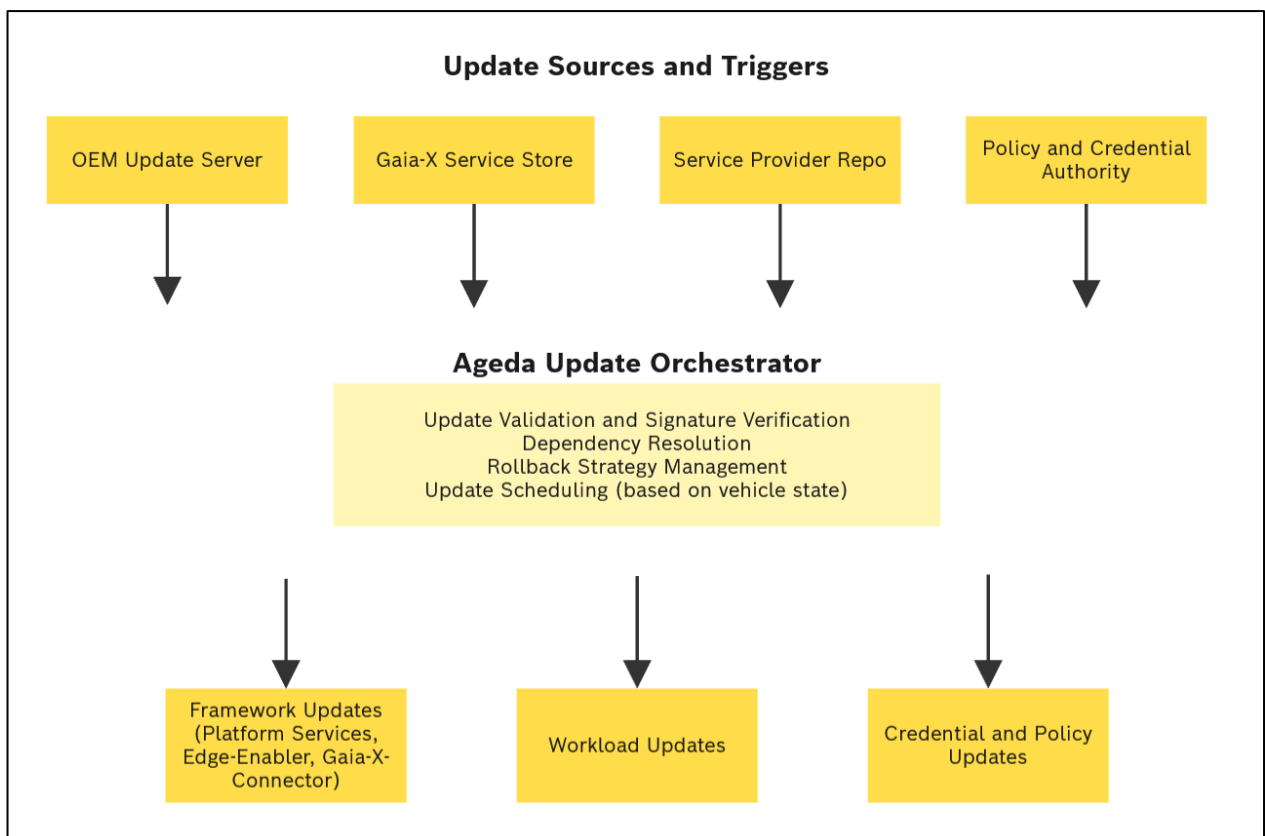


Abbildung 37: Update Konzept Umriss

Das Konzept unterscheidet vier verschiedene Auslöser für Updates. Diese aktivieren den Update-Orchestrator. Dies ist die Kernkomponente für dieses Konzept, welche die Updates validiert, Abhängigkeiten regelt, Rollbacks koordiniert und bestimmt, wann Updates ausgeführt werden. Die Funktionalität ist eine Mischung aus dem Ankaio Workload Orchestrator und der Policy Engine, somit könnten bereits entwickelte Konzepte wiederverwendet werden. Es wird zwischen drei Updates unterschieden:

AGEDA-Framework Kernkomponenten:

Für die AF-Kernkomponenten könnten A/B Partitionsupdates verwendet werden. Sobald ein Update verfügbar ist, wird die ungenutzte Partition geupdated und bei Neustart diese verwendet. Diese Updates werden von einem OEM Update Server ausgelöst. Für eine erhöhte Sicherheit werden diese Komponenten nur bei geparktem Fahrzeug durchgeführt werden. Um den Prozess zu vereinfachen, werden alle drei Komponenten (Plattform-Services, Edge-Enabler und Gaia-X-Connector) gemeinsam geupdated.

Workloads:

Updates der User-Workloads können über den Ankaio Workload-Orchestrator laufend durchgeführt werden. Die Updates werden über den Gaia-X Service Store oder ein Service Provider Repository getriggert. Nach dem Update wird das vorige Image für 24h beibehalten, falls Probleme auftreten sollten.

Credential und Policy Updates:

Die Verifiable-Credentials und wallet Konfigurationen können instantan ohne einen Neustart des AGEDA-Frameworks durchgeführt werden.

Der Ablauf eines Updates ist in der nachfolgenden Abbildung dargestellt.

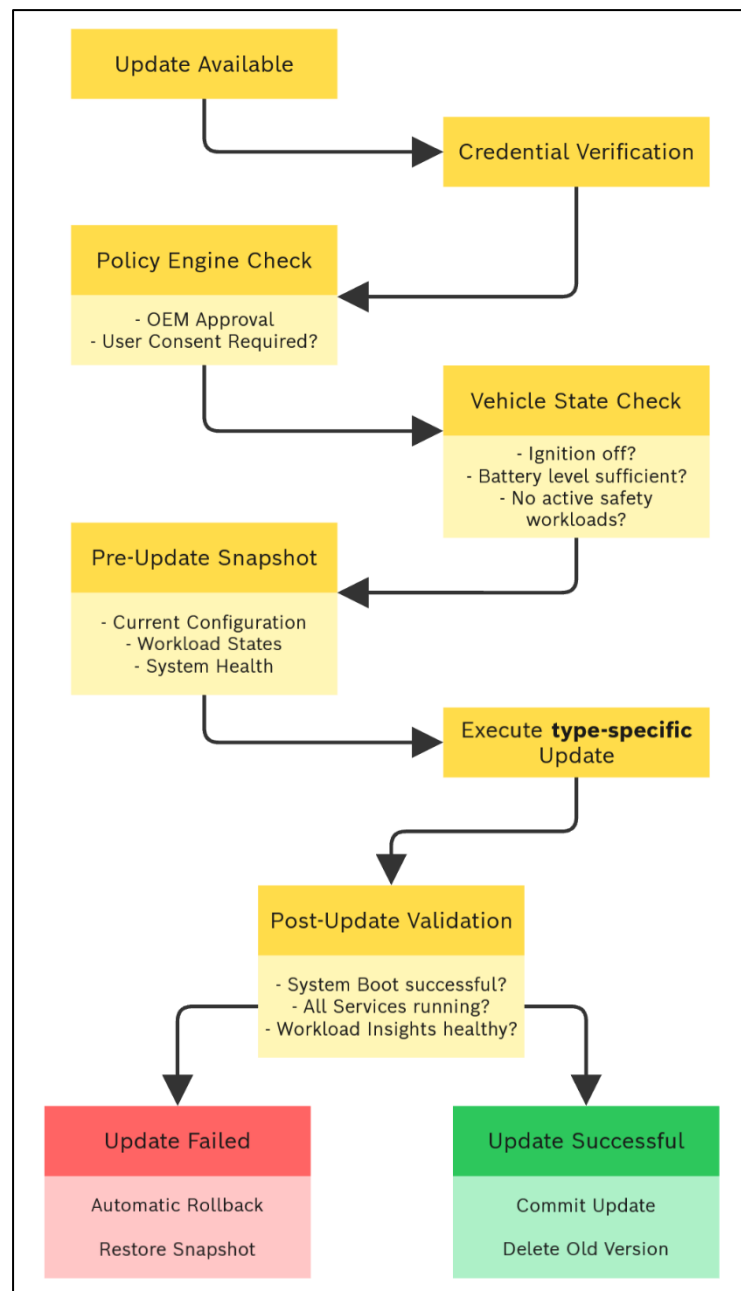


Abbildung 38: Ablauf eines Updates

Nachdem ein Update verfügbar ist, werden zuerst drei Dinge geprüft: Die Credentials, die Policy Engine und der Fahrzeugstatus. Ist hier alles okay, wird ein Schnappschuss des Systems gemacht. Dann wird, abhängig von der jeweiligen Komponente das Update durchgeführt. Sollte das Update fehlschlagen, wird das System auf den vorher erzeugten Schnappschuss zurückgesetzt. Bei einem erfolgreichen Update wird der alte Schnappschuss gelöscht.

Ein solches Konzept für das Update verschiedener Teile des AGEDA Framework bietet eine erste schlanke Grundlage. Einige Funktionalitäten sind bereits in vorhandenen Komponenten umgesetzt und somit wäre der Aufwand nicht groß.

4 Wichtigsten Positionen des zahlenmäßigen Nachweises

Die nachfolgend aufgeführten Positionen stellen die wichtigsten unmittelbaren Vorhabenkosten für das Förderprojekt AGEDA laut zahlenmäßigen Nachweises der Robert Bosch GmbH dar:

- Der größte Posten waren die Personalkosten für die Bearbeitung der Aufgaben in den drei Teilprojekten von AGEDA.
- Der zweitgrößte Posten waren sonstige unmittelbare Vorhabenkosten für das Projektbüro, welches Projektmanagementleistungen erbrachte.
- Anschließend folgen die vorhabenspezifischen Materialkosten zum Aufbau eines Versuchsfahrzeuges sowie der Laboraufbauten zur Entwicklung und zum Test des AGEDA Frameworks inkl. der Monitoring und Update Use Cases.

Die Ausgabenplanung wurde größtenteils eingehalten, größere Abweichungen sind im zahlenmäßigen Verwendungsnachweis dargestellt.

5 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die Themenfelder GAIA-X und flexible Fahrzeugarchitekturen im Software-defined Vehicle (SdV)-Kontext – insbesondere deren Umsetzung in marktfähigen Produkten – bergen in allen Teilaspekten erhebliche technisch-wissenschaftliche sowie wirtschaftliche Risiken. Diese Randbedingungen erschweren es selbst großen Automobilzulieferern, grenzenlos in die Erforschung neuer Technologien und Konzepte zu investieren. Eine öffentliche Zuwendung ist daher unerlässlich, da das Konsortium das hohe Projektrisiko nicht aus eigener Kraft tragen kann.

Ohne staatliche Förderung wären die Aufgaben des Projekts in der vorgeschlagenen Form nicht oder nur deutlich verzögert in Angriff genommen worden. Obwohl die angestrebten Ergebnisse einen hohen technologischen Wert besitzen, wäre das Projekt unter Wahrung unternehmerischer Vorsicht aufgrund des hohen Risikos in diesem Umfang nicht durchführbar gewesen. Die Gewährung von Fördermitteln ermöglichte somit eine deutlich frühere Erzielung der angestrebten Projektergebnisse sowie eine stärkere Vorantreibung von Standards und Anwendungen im GAIA-X-Umfeld.

Die im Rahmen von AGEDA durch die Robert Bosch GmbH geleisteten Arbeiten waren sowohl notwendig als auch angemessen, da sie der Planung gemäß Vorhabenbeschreibung entsprachen und zur Erfüllung der gesetzten Ziele erforderlich waren. Der Erfolg des Vorhabens wurde weiterhin in der Abschlusspräsentation, zahlreichen Demonstrationen und Publikationen belegt. Der durch die Förderung ermöglichte Austausch – beispielsweise zu GAIA-X, flexiblen Fahrzeugarchitekturen sowie Monitoring und Update – hätte in diesem Umfang sonst nicht stattfinden können.

6 Voraussichtlicher Nutzen und Verwertbarkeit der Ergebnisse

Der Beitrag der Robert Bosch GmbH zum Projekt AGEDA erfolgte durch den Geschäftsbereich Cross-Domain Computing Solutions. Die interne Arbeit an AGEDA war dabei von einer engen Zusammenarbeit zwischen der Vorausentwicklung und der Produktentwicklung innerhalb des Geschäftsbereichs sowie den Abteilungen Compute Enhanced und Compute Performance in den für AGEDA relevanten Aufgabengebieten geprägt. Der Transfer der gewonnenen Erkenntnisse in die Produkt- und Serienentwicklung erfolgte kontinuierlich.

Im Bereich der Authentifizierung und GAIA-X lieferte AGEDA wichtige Erkenntnisse für zukünftige Produktentscheidungen, wodurch die Verwertungsaufgaben erfüllt werden konnten. Insgesamt verfügt Bosch hinsichtlich der AGEDA-Ergebnisse über alle Voraussetzungen zur firmeninternen Verwertung, mit sehr hohen Erfolgsaussichten.

Weitere Informationen zum Verwertungsplan sind im nichtöffentlichen Erfolgskontrollbericht aufgeführt.

7 Erfolgte oder geplante Ergebnisverbreitung

Während der Projektlaufzeit beteiligte sich Bosch aktiv an der Erstellung von Präsentationsmaterialien für diverse Veranstaltungen und an der Verfassung von Whitepapern. Darüber hinaus wird die Möglichkeit geprüft, die entwickelten Software-Komponenten vollständig oder teilweise als Open Source zu veröffentlichen.

8 Literaturverzeichnis

- [1] Böhlen, Fischer: Brückenschlag zwischen Fahrzeugdiagnose und Datenzugriff; 17. Tagung Diagnose in mechatronischen Fahrzeugsystemen, Dresden, 2025
- [2] Röper, C., Buch D., Fiedler, J., Ziesig, S. (2024): Modularisierung eines SOVD-Servers für vernetzte und Software-definierte Fahrzeuge, 17. Tagung Diagnose in mechatronischen Fahrzeugsystemen, Dresden
- [3] <https://confluence.4373.aec.auence.com/confluence/display/AGEDA/Architecture+Decision+Record>